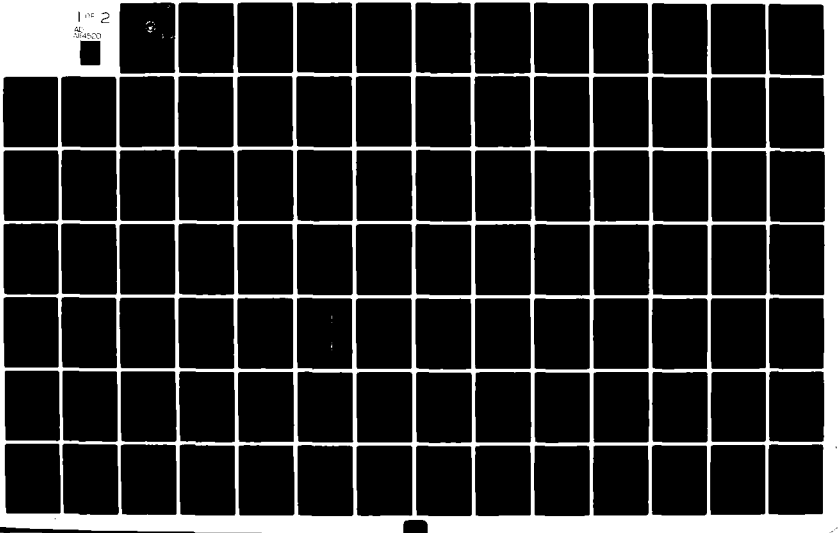
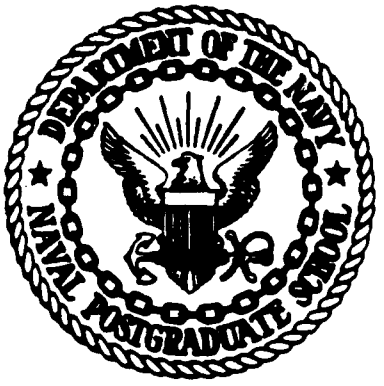
AD-A114 520 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 9/2
A MACRO APPROACH TO SOFTWARE RESOURCE ESTIMATION AND LIFE CYCLE--ETC(U)
DEC 81 B R VORGANG

UNCLASSIFIED NL

1 OF 2
AD
A114520



NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

A MACRO APPROACH TO SOFTWARE RESOURCE ESTIMATION AND LIFE CYCLE CONTROL

by

Blair Roland Vorgang

December 1981

Thesis Advisor: N. R. Lyons

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A114520	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Macro Approach to Software Resource Estimation and Life Cycle Control	5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; December 1981
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Blair Roland Vorgang	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	12. REPORT DATE December 1981
	13. NUMBER OF PAGES 144
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software Development; Software Economics; Software Engineering; Life Cycle Management; Life Cycle Cost; Information System; Software Cost Estimating; Life Cycle Control

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Planning and controlling the software development process has shown, in the past, to be an extremely difficult task. The estimation of resource requirements, development costs, risk profiles and project feasibility has often proven to be inaccurate, thus costing the government time and dollars. However, by using obtainable management parameters, and simple engineering and operations research techniques, estimating can be done easily and accurately by taking a macro approach to the estimation problem.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

This study will present the background and mathematical basis for a software cost estimation model. In addition, an example of an automated application of the model will be presented and discussed.

Approved for public release; distribution unlimited

A Macro Approach to Software Resource
Estimation and Life Cycle Control

by

Blair Roland Vorgang
Captain, United States Marine Corps
B.S., United States Naval Academy, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
December 1981

Author: Blair R. Vorgang

Approved by: Norman R. Lyons
Thesis Advisor

M. L. O'Neill
Co-Advisor

Chairman, Department of Administrative Sciences

Willis R. Greer, Jr.
Dean of Information and Policy Sciences

ABSTRACT

Planning and controlling the software development process has shown, in the past, to be an extremely diffcult task. The estimation of resource requirements, development costs, risk profiles and project feasibility has often proven to be inaccurate, thus costing the government time and dollars. However, by using obtainable management parameters, and simple engineering and operations research techniques, estimating can be done easily and accurately by taking a macro approach to the estimation problem.

This study will present the background and mathematical basis for a software cost estimation model. In addition, an example of an automated application of the model will be presented and discussed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

I would like to express my thanks to Mr. Lawrence H. Putnam for his invaluable assistance; for without his help, this study would not have been possible.

I. INTRODUCTION

A. GENERAL

Software development within the Federal Government has been significantly marred by a history of projects that are characterized by cost and schedule overruns, and a delivered end-product that fails to provide the desired performance and function originally required. This statement is not mere conjecture, but has been observed and documented in the past. The compilation of responses to a questionnaire sent out by the United States General Accounting Office (GAO) to one hundred and sixty three software contracting firms and one hundred and thirteen experienced federal project officers has ascertained that in over 50 percent of the cases, 62.0 percent of the respondents experienced software development calender overruns, 50.4 percent of the respondents experienced software cost overruns, and 43.3 percent of the respondents received software that required substantial in-house correction or modification prior to being effectively used. [Ref. 26: 8-10]

In an industry in which the Federal Government incurs expenditures and fiscal obligations well into the billions of dollars, the process of resource estimation and project control for software development is a major budgetary concern. As development costs steadily and rapidly escalate, this

concern gains added complexity (Figure 1) and some manner of formal methodology which allows the project manager to estimate the resource requirements and to control the project becomes necessary.

The requirements of such a methodology must allow for the capability to operate with any size project yet still remain a function of manageable parameters such as effort and development time. These parameters require relationships to the system size (delivered source lines of code) in terms of such environmental factors as:

system complexity
use of and type of software engineering tools
user interface
use of a target machine
use of a development computer
the development language used
other human factors
[Ref. 20: 14]

In addition, the concept of system difficulty is a factor that must be taken into account, not only in terms of the number and types of source lines of code, but also in terms of the development approach and the integration of modules and subprograms.

The Department of Defense (DOD) has taken steps to formalize and standardize the governing of the life-cycle of automated information systems (AIS). Life cycle management

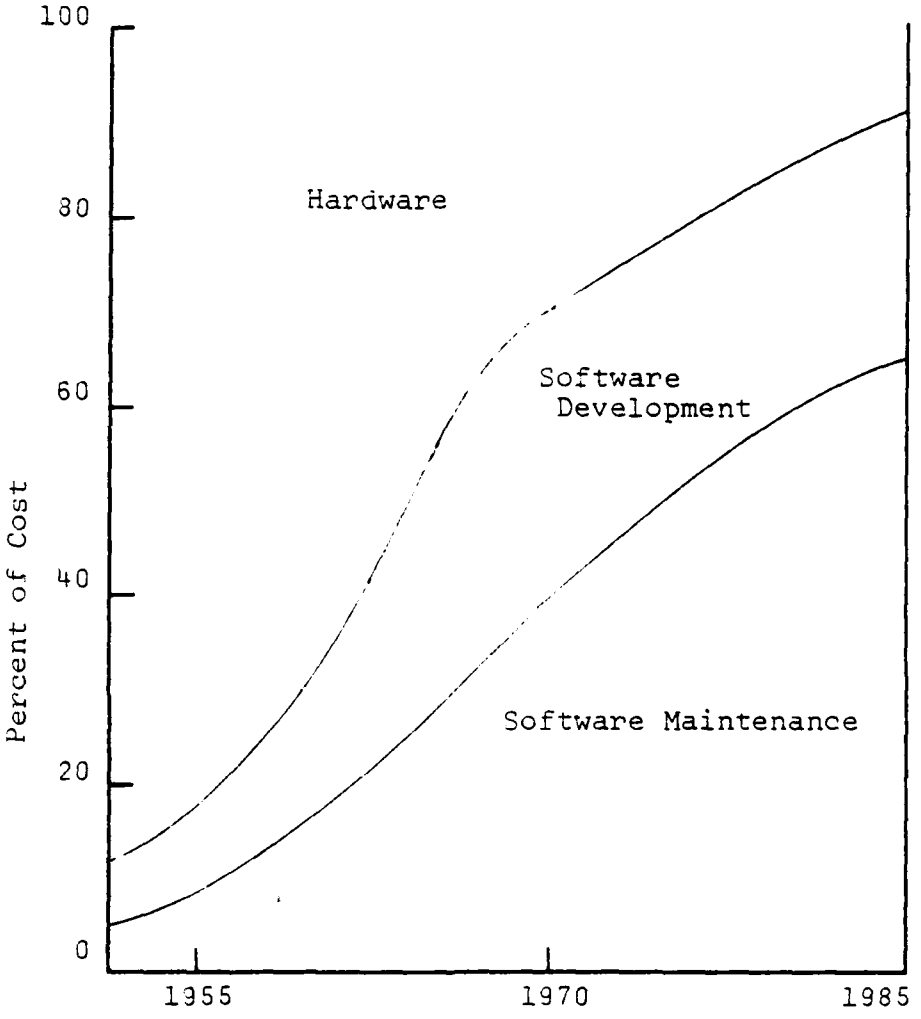


Figure 1

Hardware versus Software Cost Trends
[Ref. 3: 1227]

(LCM), as defined by the Department of Defense, is "the process for administering an AIS over its whole life with emphasis on strengthening early decisions which shape AIS costs and utility." [Ref. 6: 2]

Among the objectives of life cycle management is to assure the lowest total overall cost and to provide visibility for resource requirements. In order that a project manager might plan and control a software development project throughout the entire life-cycle, he must be able to answer four simple management questions in terms of cost and resources:

Is the project feasible?

What are the resource requirements?

How long will it take?

What are the risks?

Unfortunately, the past has shown that these simple questions can prove to be extremely difficult to answer. Not only must these questions be answered during the initial stage of the project, but also throughout the entire life of the project. "Software development is dynamic...not static." [Ref. 20: 181]

B. SCOPE

This study is directed toward the role of the manager, his ability to address and answer the four management questions, and the role of these questions in life-cycle control. Much of the background information is technical/mathematical in nature. These topics will be presented in such a manner

as to allow the reader to grasp the fundamental concepts without unnecessary inundation by the mathematics involved. The emphasis will remain on the conceptual aspects and in the resultant numerical answers.

The principles presented in this study are applicable to software development project managers both inside and outside the federal government. Because the Department of Defense is one of the largest single users of computer technology, this study will reflect Department of Defense policies and techniques.

C. ASSUMPTIONS

It is assumed that the reader has a working understanding of life cycle management and the software development process.

D. ORGANIZATION OF STUDY

This study will be organized in such a manner as to flow from the conceptual to the specific. After the presentation of necessary background material in Chapter II, greater specification occurs in Chapter III in dealing with the Rayleigh curve, its characteristics, and its relationship to the life-cycle. Chapter IV will deal with the characteristics and application of the Putnam Software Equation. Chapter V will present computerized applications of this methodology using the SLIM (Software Life Cycle Model) software estimation package. This study relies heavily on the writings and observations of Lawrence H. Putnam, President of Quantitative Software Management, Inc.

E. SUMMARY

This study will present a means of answering the four basic management questions concerned with the control and planning of software development:

Is the project feasible?

What are the resource requirements?

How long will it take?

What are the risks?

This will be accomplished in a manner which uses the basic management parameters of time, cost, and manpower, as the inputs for planning and control.

II. BACKGROUND

A. AN OVERVIEW OF LIFE-CYCLE MANAGEMENT AND THE SOFTWARE DEVELOPMENT PROCESS

1. Life-Cycle Management

The Department of Defense subdivides the life-cycle of an automated information system into five broad phases:

Mission Analysis/Project Initiation

Concept /Development

Definition/Design

System Development

Deployment/Operation

[Refs. 6: 2, 10: para 1003.3]

These phases are sequential and act as a control mechanism for both systems development and management accountability.

a. Mission Analysis/Project Initiation

This phase serves to identify a mission element need in terms of functional requirements, validate the need, and recommend the exploration of alternate functional concepts that satisfy the need. Although this particular structure exists in large projects requiring extensive resource commitments, the philosophy of mission need can be persuasive in smaller programs and should be carried out and documented in a like manner. This phase is completed at Milestone 0, the approval of the Mission Element Need Statement (MENS).

b. Concept Development

The Concept Development phase serves to define requirements, evaluate the alternative methods that satisfy the need, and to recommend one or more feasible concepts for further exploration. This phase terminates with approval for the vendor to demonstrate the alternative concepts or to proceed directly to the definition and design phase, given that one concept has been selected. Termination of this phase indicates Milestone I.

c. Definition/Design

The purpose of the Definition/Design phase is to fully define the functional requirements in terms of system/subsystem specifications, and to design an operable system. This phase terminates at Milestone II when both the definition and design concepts have been approved.

d. System Development

The System Development phase serves to develop, integrate, test and evaluate the system. This phase is completed when the appropriate functional officials grant approval, certifying that the system satisfies the mission need. Approval constitutes Milestone III.

e. Deployment and Operation

This phase covers the implementation of the approved system, the continued operation of the system, and all required maintenance and modification.

2. The Software Development Process

There are several ways of viewing the software development process. For purposes of this study, the software development process will consist of four phases:

Systems Definition

Functional Design/Specification

Development

Operation and Maintenance

The phases are essentially sequential, but some degree of overlap can be permitted. In addition to the four phases, there is one step, Installation, which overlaps both the Development phase and the Operation and Maintenance phase (Figure 2).

a. Systems Definition

Systems definition includes the complete definition of the preferred alternatives or alternative, the establishment of bounds on manpower effort and development time, and the refinement of costs.

b. Functional Design/Specification

This phase involves the preparation of detailed system specifications for both the application and technical software support, and the finalization of technical procedures and programming policies. Finally, detailed system specifications are converted into detailed programming specifications.

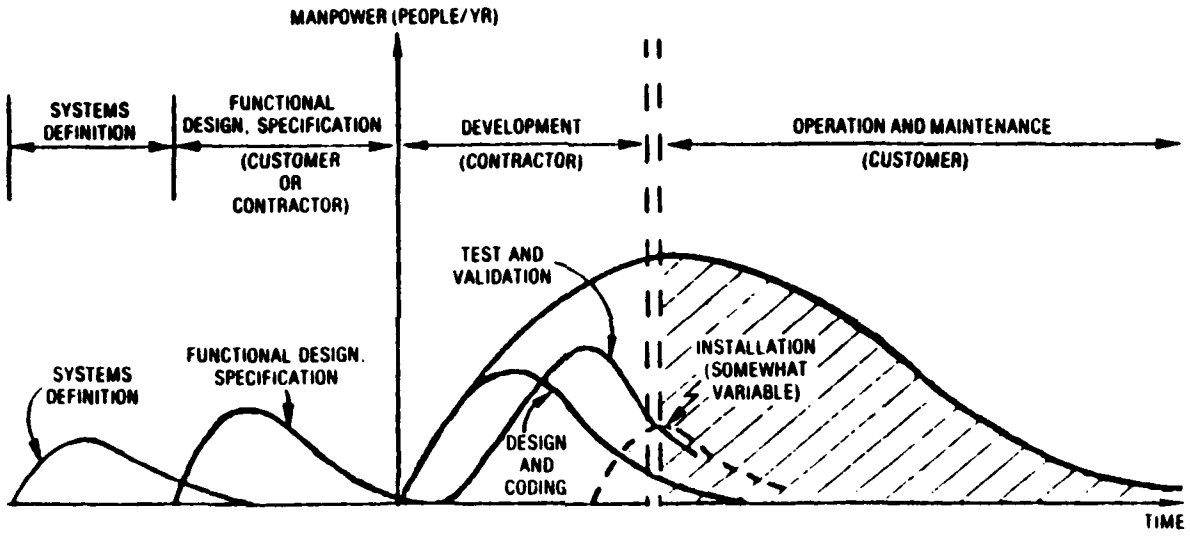


Figure 2

The Software Life-Cycle
[Ref. 20: 15]

c. Development

The Development phase can be further subdivided into two steps:

Design and Coding

Test and Validation

These two steps include the coding of applications specifications and the conducting of unit and string testing of the machine executable instructions. In addition, program documentation is completed. System testing requirements are prepared and the test is carried out.

d. Operation and Maintenance

The final phase, Operation and Maintenance, includes system refinement, tuning, post-implementation reviews, and the continued operation and required maintenance.

e. Installation

The installation of the system overlaps the end of the Development phase and the beginning of the Operation and Maintenance phase. This step includes implementation and conversion planning as well as the actual conversion and phased implementation. [Ref. 2: 14-18]

B. MICRO VERSUS MACRO METHODOLOGY

The traditional approach to size/time/cost estimation has been the use of a micro-methodology. (Table 1) This methodology is largely intuitive and begins by fixing the

Table 1

Traditional Methods of Cost Estimation
[Refs. 1: 229-231, 11: 24, 27: 12-13, 29: 618-619]

1. Experience Method
2. Constraint Method
3. Top-Down Estimating
4. Similarities and Differences Estimating
5. Analogy Method
6. Standards Estimating
7. Ratio Estimating
8. Bottom-Up Estimating
9. Units of Work Method
10. Smoothing and Extrapolation Estimating
11. *Number of Instructions*
12. Quantitative Method
13. Cost Per Instruction
14. Percent of Hardware Method

size, start date and duration of each distinguishable activity. Adjustments are estimated and applied based on the caliber of personnel, complexity, and so forth. Finally, activities are arranged using network models such as PERT (Program Evaluation and Review Technique) and CPM (Critical Path Method). [Ref. 11: 23] To even further quantify the development process, the number of modules within the entire process, the number of statements per module, and the cost per statement are estimated. [Ref. 20: 13] Productivity is the critical input to this methodology. The major drawback is, however, that productivity varies greatly between organizations and individuals.

The micro approach can work well on small projects where there is little detail involved and few programmers are required. Troubles develop when the program being developed involves large volumes of detailed specifications and hundreds of thousands of lines of source code.

When using traditional methodologies, most errors in cost estimation are a result of underestimating size, sometimes by as much as a factor of three. This is usually the result of erroneously relating productivity to delivered source lines of code. In addition, it is common to find a productivity factor that has a standard deviation two and one half times greater than the expected value. [Ref. 7: 2] More discussion concerning productivity will be accomplished later in this chapter as well as in Chapter IV.

The macro approach views the project from a higher level. Beginning as early as the Systems Definition phase, management parameters serve as inputs to the methodology. The inputs can be used to generate the expected life-cycle curve of manpower versus time. As the project progresses and the parameters become firmly established, the life-cycle curve can be updated. Milestones can be located along the curve and parallel tolerance curves can be generated to indicate the percentage of uncertainty. Through the information presented along the life-cycle curve, control can be exercised over the life-cycle of the project. The key to this approach is that the curve can be shown in terms of management parameters: manpower, time, and effort. [Refs. 11: 24, 20: 130] This is the approach that will be presented in this study.

C. SOFTWARE MYTHS

Most large-scale software development projects that go astray do so because of calender overruns. [Refs. 5: 14, 26: 9] Much of the problem with overruns can be traced to myths that surround software engineering:

Development effort is the product of people and time.

Programmer productivity (source statements per unit of work) is constant and can be set by management.

Productivity varies directly with the effort applied.

Men and months (time) are interchangeable.

[Refs. 5: 14, 16: 352, 17: 348, 20: 14]

1. Development Effort

The assumption that development effort scales linearly as the product of people and time (Figure 3) leads one to believe that time can be arbitrarily set by management and that the requisite number of people can be assigned to achieve the desired results. As will be shown in the following chapter, this reasoning can lead to both calender and cost overruns.

2. Productivity

Traditionally, management estimates the number of source lines of code and applies a historical overall productivity rate to determine a man-year number. For example:

Given:

Source Lines of Code (Ss) = 200,000 lines of code (estimate)

$$\text{Productivity (PR) (historical)} = \frac{\text{Total Source Lines of Code}}{\text{Total Effort}}$$

$$= 1000 \text{ Ss/MY (man-years)}$$

therefore:

$$\text{Development Effort (E)} = \frac{200{,}000 \text{ Ss}}{1000 \text{ Ss/MY}} = 200 \text{ MY}$$

Lawrence Putnam, during the course of his studies, performed a least squares best fit, using data taken from over four hundred projects of the United States Air Force's Rome Air Development Center, against delivered source lines of code for each of the four hundred projects. The data

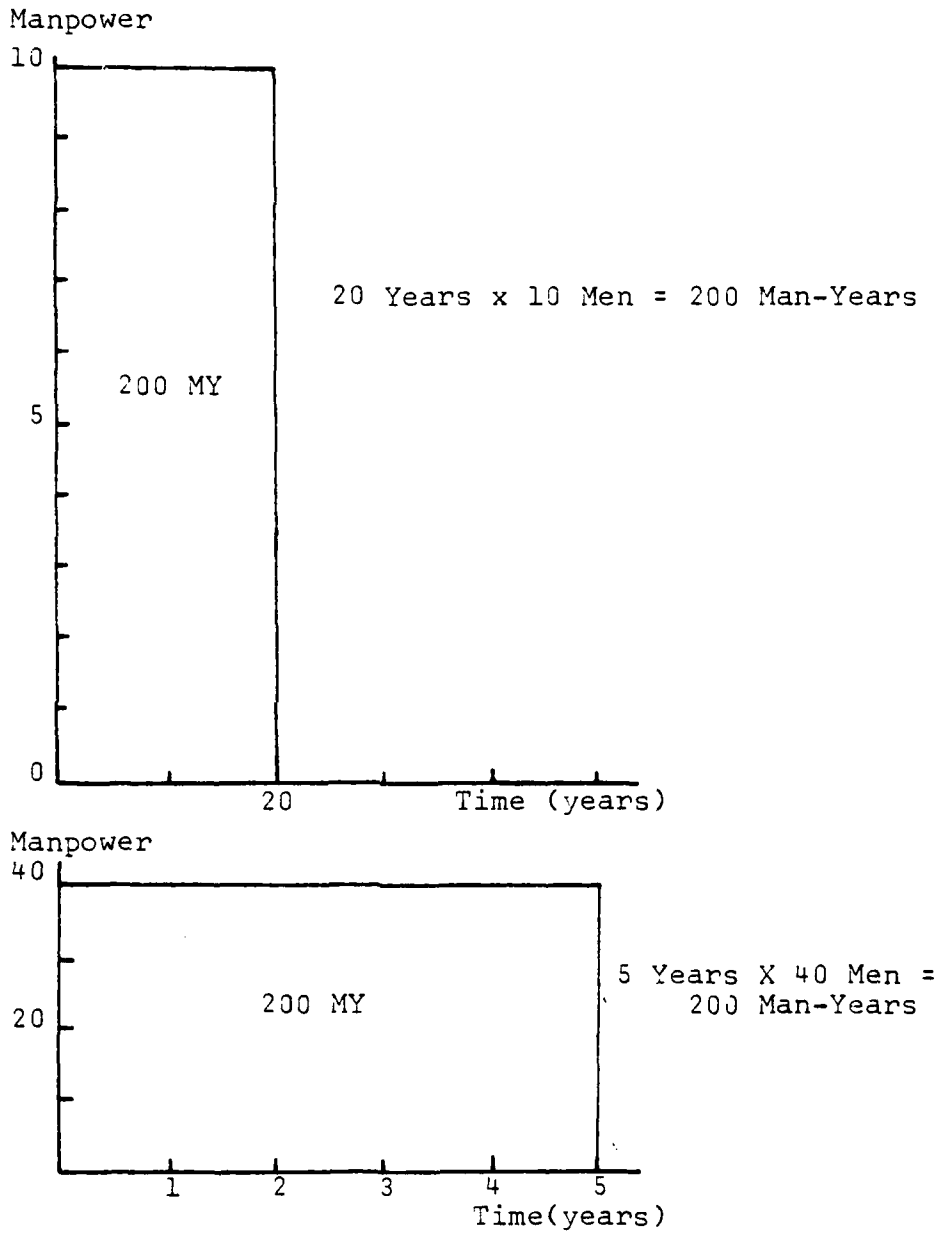


Figure 3

The Development Effort Myth

expressed a wide range in system size (100 to 1,000,000+ source lines of code), project duration (one month to 6 years+), man-months of effort (one MM to 20,000 MM), average number of people (one to several hundred), and productivity (10 Ss/MM to several thousand Ss/MM). The resulting correlation coefficient after performing a least squares best fit to productivity (Ss/E) versus delivered lines of source code was found to be r = .033415, thus displaying virtually no correlation. [Refs. 20: 29-30, 24: 87]

However, after performing further least squares best fit relations, project duration in months versus delivered source lines of code showed r = .700256. When total man-months versus delivered source lines of code were predicted linearly, r was shown to equal .853915. When a least squares best fit was performed for the average number of people involved in the development effort versus delivered source lines of code, the correlation coefficient showed r = .80388. [Refs. 29: 29-30, 24: 88-90] Although standard deviations proved to be large, three of the variables did show good correlation to delivered source lines of code. Furthermore, it was shown that there is virtually no correlation between productivity and delivered source lines of code (Table 2).

C. E. Walston and C.P. Felix, in their study, began a software measurement project for the IBM (International Business Machines Corporation) Federal Systems Division in

Table 2

Correlation Coefficients (r)

(Based on Least Squares Best Fit against Delivered Source Lines of Code (DSLOC))

	Correlation Coefficient (r)
Productivity (Ss/E)	0.033415
Project Duration Time	0.700256
Total Effort (MM or MY)	0.853915
Ave. Number of People Involved in Development Effort	0.80388

1972. They found twenty-nine variables that had an extremely high correlation with productivity. [Ref. 28: 62] The figures concerning these variables are tabulated in Appendix A.

3. The Interchangeability of Men and Months

One of the largest problems associated with software engineering evolves from the erroneous utilization of the concept of the man-month. Because cost varies with the product of men and the number of months (time), using man-month as a means of measuring the size of a job implies that men and months are interchangeable. Unfortunately, they are not. Man-month is a measure of effort. "The number of months of a project depends upon its sequential constraints. The maximum number of men depends upon the number of independent subtasks." [Ref. 5: 25-26]

If an appropriate development schedule is established at the outset of a project, satisfactory work can be accomplished within the allotted time by the assigned programmers. [Ref. 11: 23] Otherwise, the oft quoted Brooks' Law takes over: "Adding manpower to a late software project makes it later." [Ref. 5: 25]

D. SUMMARY

Life-cycle management and the software development process offer a structured means for planning, developing and controlling the software project. Traditionally, the resource

estimation process has been a micro approach. This is an intuitive approach which hinges extensively on the relationship between productivity and delivered source lines of code. Data from past projects indicates that no correlation exists between the two. Another erroneous assumption made in the software engineering field is the use of man-month as a measure of the size of a project. Man-month is a measure of effort. Another estimating approach, macro-estimating, makes use of accessable management parameters: time, manpower, and cost. In order to accurately estimate the project resource requirements, the various software myths that manifest themselves in the more traditional approaches have to be dismissed.

III. THE LIFE-CYCLE MANPOWER CURVE

As previously shown, the software development process can be subdivided into sequential and overlapping phases. These phases can also be further subdivided into steps. The relationship between the phases within the system development process is such that required work for a particular phase cannot begin until specific work in an earlier phase is completed. The phases display technological interdependence. [Ref. 13: 156]

The software development process also displays the traits of a homogeneous project. The development process is such that each phase has at least one technological interdependence tie to another phase. Otherwise, each phase could be considered an independent process. [Ref. 13: 156]

A. CHARACTERISTICS OF THE RAYLEIGH MODEL

In the life-cycle model developed by Peter V. Norden during a course of study he conducted between 1956 and 1964 at the IBM Development Laboratories in Poughkeepsie, New York [Ref. 12: 219], a small number of successive phases of work, with each phase being based upon the manner in which complex technological problems are approached, are mathematically fitted to a life-cycle curve. This life-cycle curve can be mathematically represented by the Rayleigh

equation, a mathematical form from the Weibull family of reliability functions. The equation that describes each phase is:

$$y' = 2Kate^{-at^2} \qquad \text{(Figure 4)}$$

where

y' = the manpower utilized during each time period (man-years/year or man-months/month) or the number of people involved in the activity at any given instant in time.

K = the total cumulative manpower (life-cycle effort) utilized upon completion of the project (man-years or man-months).

a = the shape parameter. This governs the time required to reach peak manpower.

t = the elapsed time in years or months, from the start of the cycle.

[Ref. 13: 156-157]

The Rayleigh curve appears to retain its validity only for an activity which requires the making of large decisions. A combination of phases will not result in a single overall Rayleigh curve. [Ref. 14: 13] This principle states that the summation of a series of overlapping Rayleigh curves will not produce another Rayleigh curve. There is, however, a significant observation that is termed the Software Development

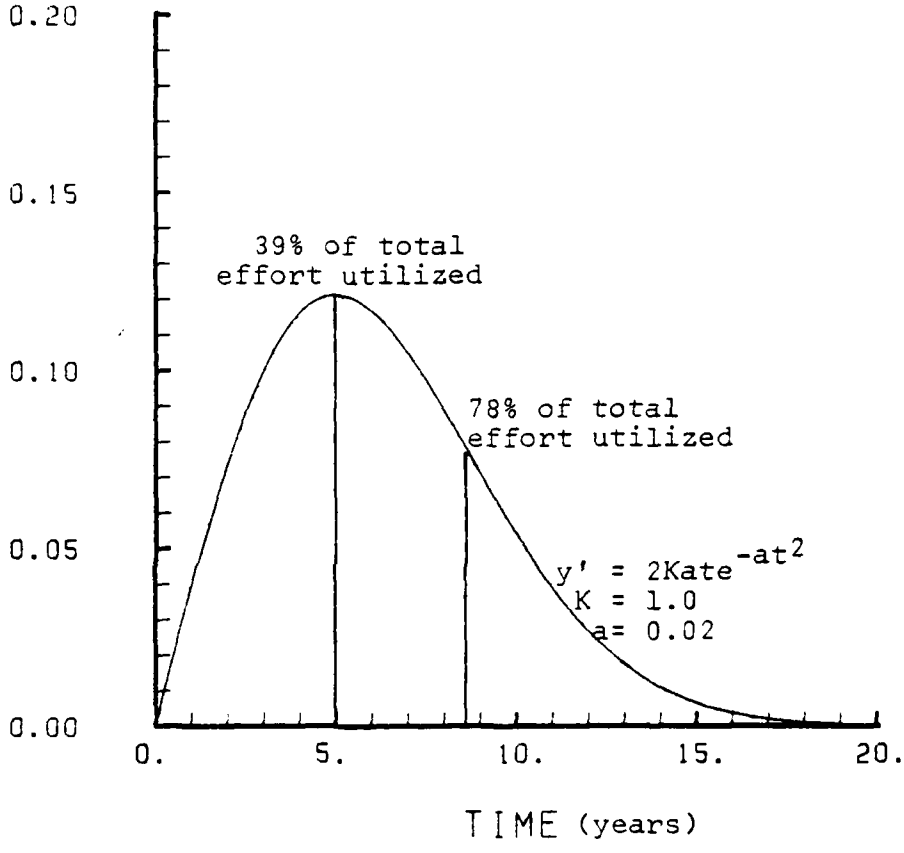


Figure 4

Current Manpower Utilization
[Ref. 13: 158]

'Single Cycle' Phenomenon that ignores this additive principle. The phenomenon is such that the process of software development yields cycles that do add up to a Weibull shape. This seems to imply that the software development process is a homogeneous problem solving effort, even though the effort is broken down into phases. [Ref. 12: 225] This phenomenon was first noticed by Lawrence Putnam, then with the U.S. Army Computer Systems Command. Its validity has since been corroborated by many other studies (Table 3). Further mention of the Rayleigh equation/curve will be in the context of software development.

Multiplying the Rayleigh equation by the labor rate converts it to a cost function. Integrating the equation over time (Figure 5) produces cumulative effort and cost of any time, [Refs. 13: 158, 20: 5]

$$y = K(1 - e^{-at^2})$$

and taking the derivitive of the Rayleigh equation (Figure 6)

$$y'' = 2Kae^{-at^2} (1 - 2at^2)$$

yields the change in the total effort at any time. [Ref. 13:158]

The Rayleigh equation is expressed in terms of management parameters, in this case, time and effort, which are necessary for a macro-methodology. These management parameters can be further expressed in terms of manloading and cash flow rates,

Table 3

A Sampling of Those Who Have
Found Evidence of Rayleigh-Like Behavior
[Ref. 20: 32-33]

Investigator	Application
50 Army Computer Systems Command (CSC) Systems	Business
Apollo-Gemini	Very Large Real-Time
NSA System	Mixed (10-100 MY)
Electronics Systems Division (ESD), USAF	Imbedded (C^3, Wpns Sys)
Defense Log Mgt Agency	Business (Logistics)
Hughes Aircraft	Mixed
GTE	Real-Time Telecomm
Apollo-Draper Labs	On Board Software
So. Cal. Edison	Small On-Line Customer Information System
General Electric Factory Control (MICS)	Medium-Size Process Control System
Wolverton's Data	Large Aerospace and C^3
Joel Aron and IBM Yorktown	Large System Software and Application
Riordan's Dynamic Model	General Systems

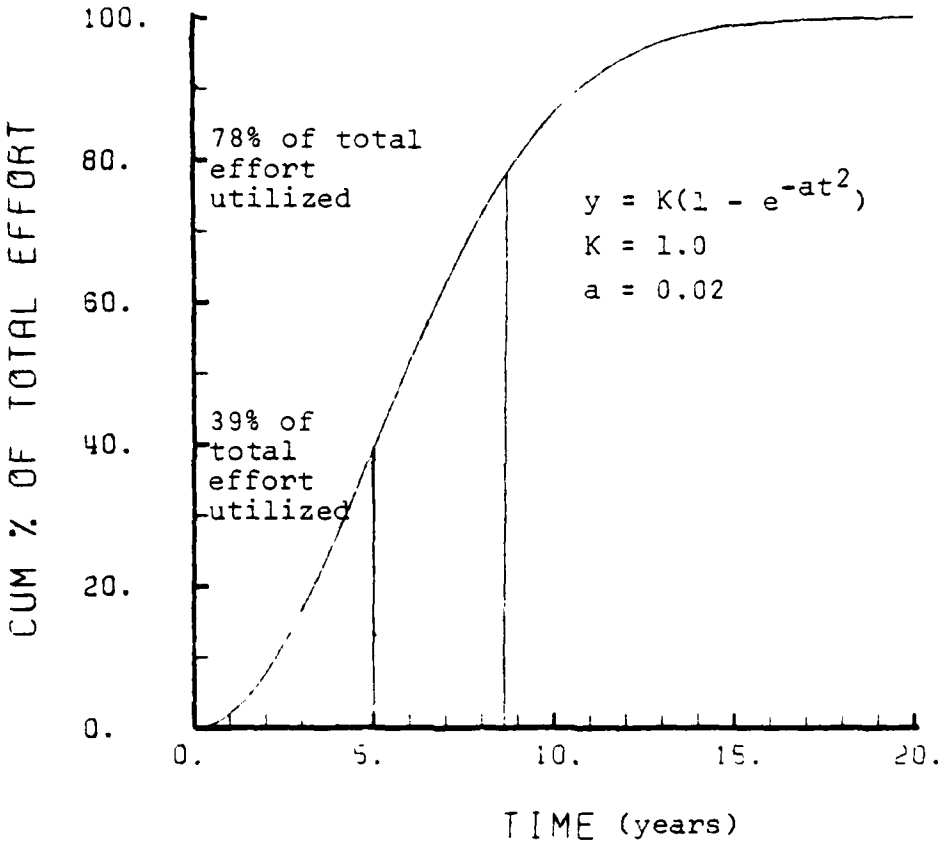


Figure 5

Cumulative Manpower Utilization
[Ref. 13: 158]

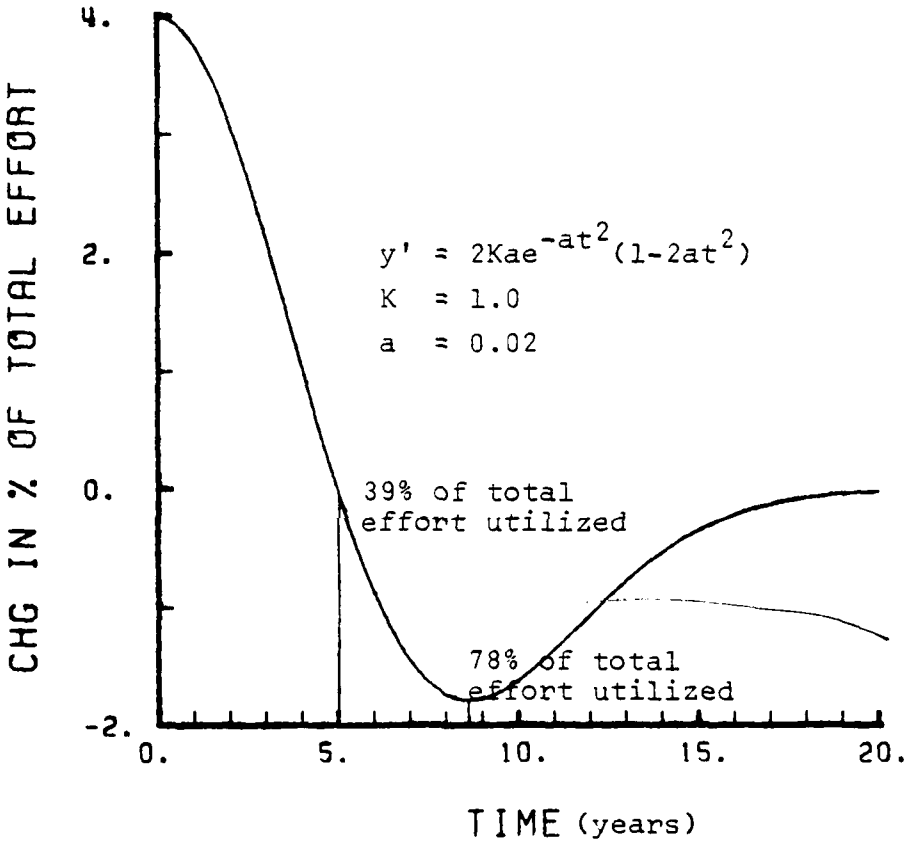


Figure 6

Change in Manpower Utilized
[Ref. 13: 158]

and cumulative effort and cost. In addition, project delivery schedules, in terms of project milestones, can be empirically located on the curve.

It has been determined that the shape parameter, 'a', is related to software development time such that

$$a = \frac{1}{2\ t_d^{\ 2}}$$

where

t_d = development time

Mathematically, t_d is the time that the Rayleigh curve reaches a maximum. It has been empirically demonstrated that this is essentially the time that a system becomes operational, the development time. [Refs. 17: 352-A, 20: 17] For purposes of this study, it will be assumed that t_d equals the development time for a system.

The relationship between development time, t_d, and the Rayleigh equation, and cumulative manpower, K, and the Rayleigh equation are graphically demonstrated in Figure 7 and Figure 8.

It is also interesting to note that, in Figure 5, thirty-nine percent of the total effort utilized takes place during the first quarter of the phase. Seventy-eight percent of the total effort utilized occurs during the first 43.5 percent of the phase. These figures point out the realistic time and

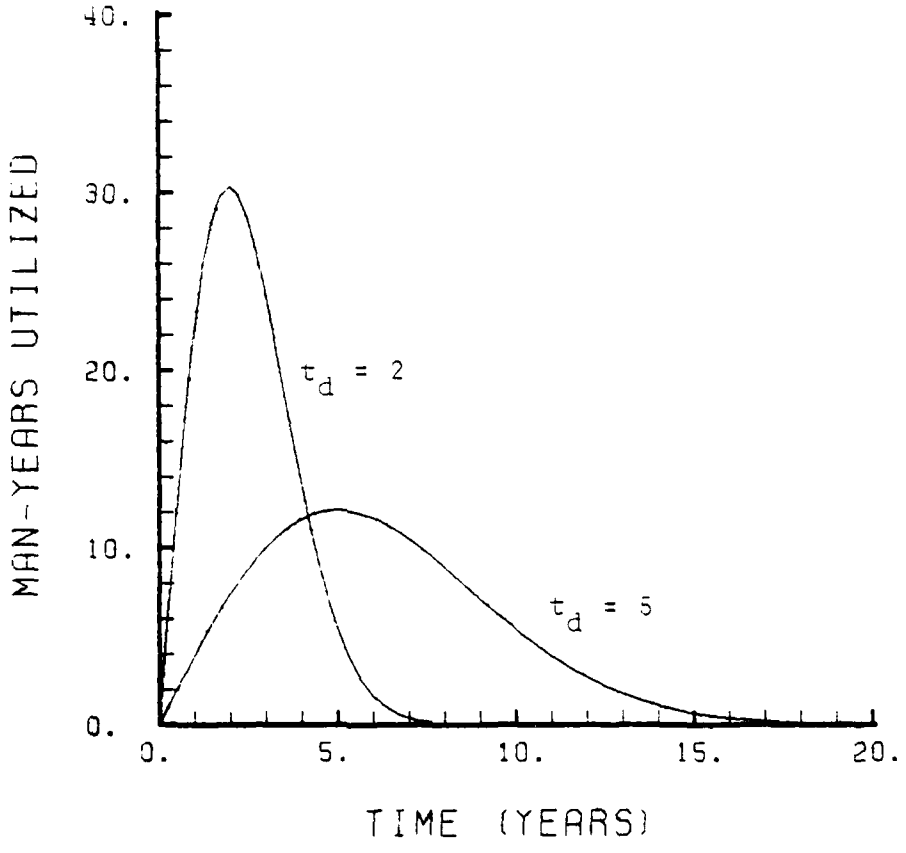


Figure 7

Change in Time-To-Peak versus Constant Manpower
[Ref. 13: 159]

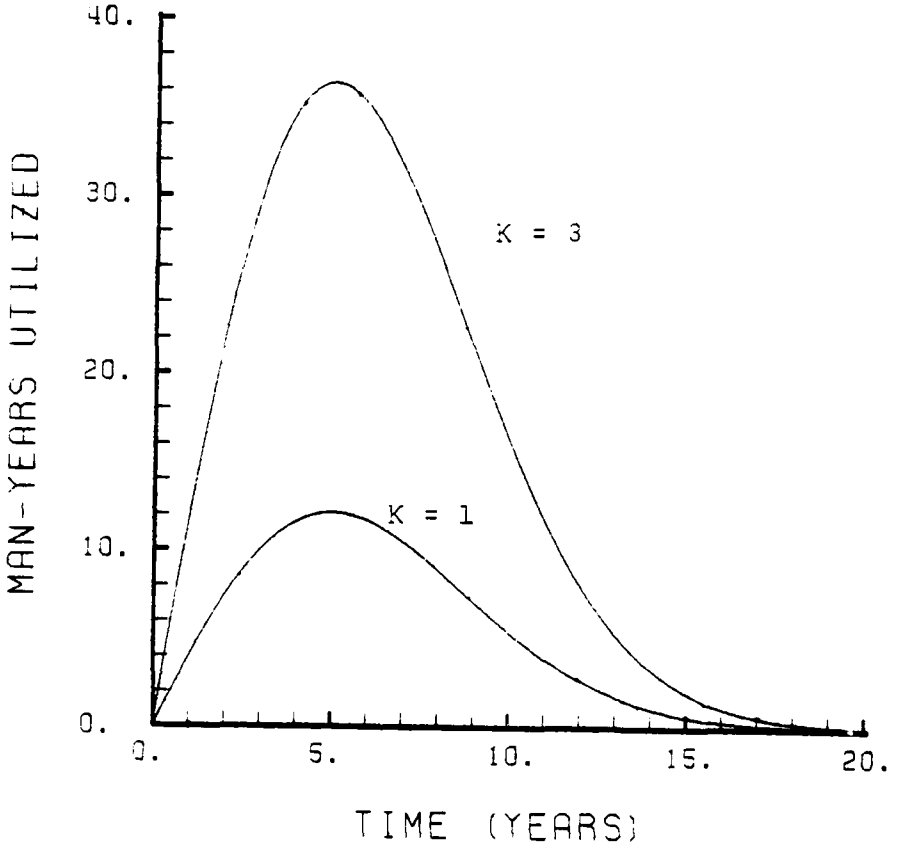


Figure 8

Change in Manpower versus Constant Time-To-Peak
[Ref. 13: 159]

manpower requirements after development time for such factors as maintenance, modifications, and continued program management.

B. EFFECT OF SYSTEM NOISE AND RANDOM BEHAVIOR

One hundred and seventy-four time history data points of thirty eight different systems, taken primarily from the U.S. Army Computer Systems Command, National Security Agency, and IBM - NASA, were normalized, in order to make the ranges comparable, and plotted. Lawrence Putnam fitted the best Rayleigh curve to the data, along with a ninety percent confidence interval, with the resulting coefficient of determination, r^2, equal to .7744 (Figure 9).

It is quite evident that there is considerable scatter in the data. One must consider that the process is subject to the laws of probability; due to less than perfect management, crises will appear and the size and solution become the probabilistic factor. In addition, it can be reasonably assumed that management did not respond to project requirements. Also, data is not necessarily recorded in a careful, precise manner. [Refs. 20: 19, 24: 74]

What allows the Rayleigh equation to be such a powerful management tool is that, even with considerable scatter, development time can be determined with an uncertainty less than three percent and that total effort can be determined

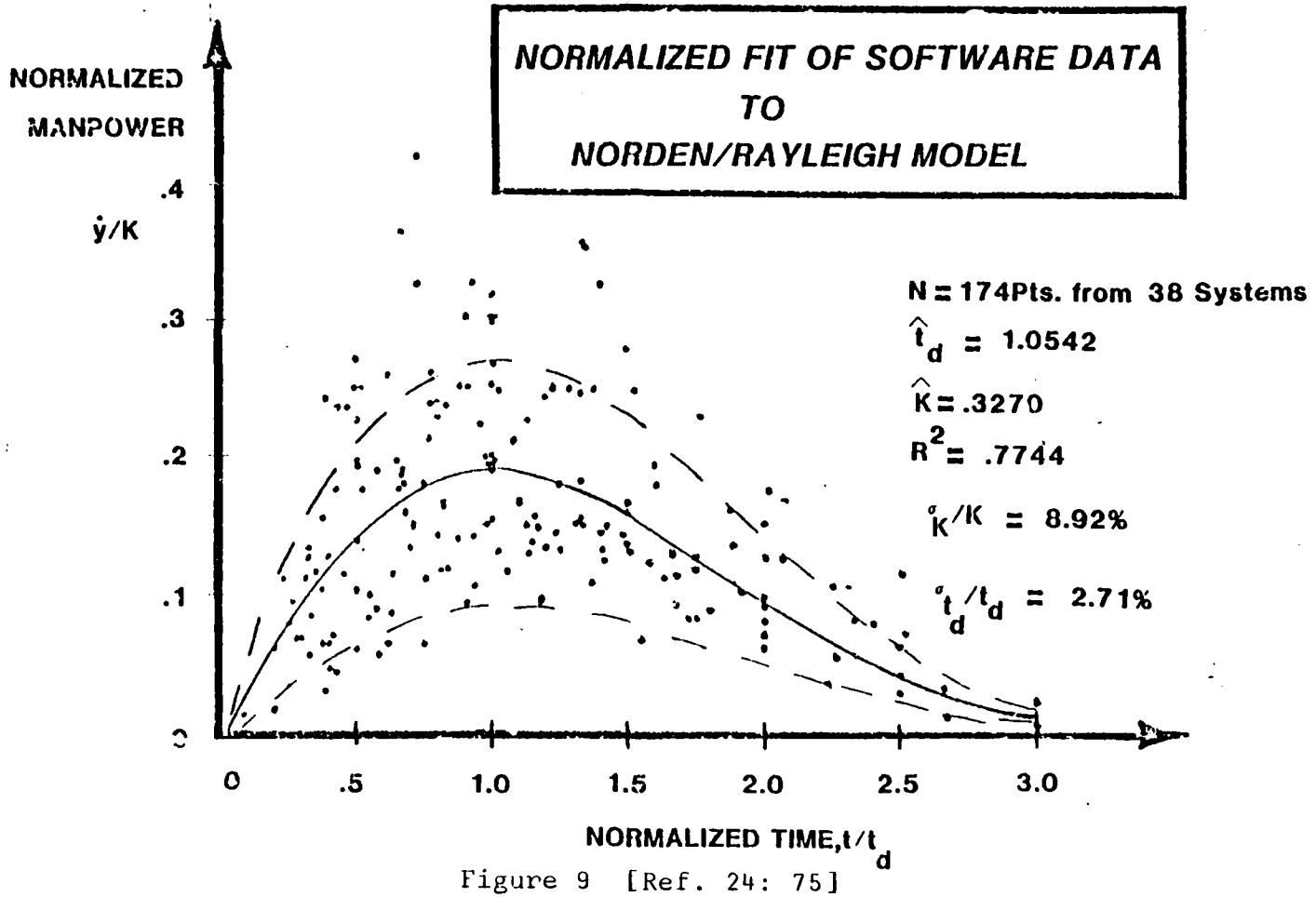


Figure 9 [Ref. 24: 75]

with less than nine percent uncertainty. It is now obvious that despite noise in the system, management parameters can be accurately determined.

C. DIFFICULTY CONCEPTS

The ratio k/t_d^2 demonstrates a significant relationship to the Rayleigh equation. The variable dimensions of this ratio reflect force, the time rate of change of momentum. [Refs. 16: 350, 22: 19] By linearizing the Rayleigh equation through the process of logarithms

$$\text{Log}(y'/t) = \text{Log}(K/t_d^2) + \frac{(-\text{Log } e)t^2}{2t_d^2}$$

and plotting the equation in terms of manpower applied to a system over time squared, with $\text{Log}(K/t_d^2)$ being the intercept and $1/(2\ t_d^2)$ the slope, a straight line is produced. Putnam performed this operation for some one hundred systems and found that the argument of the intercept, K/t_d^2, reflected system difficulty. Those systems considered easy to develop had low intercept values and, conversely, those considered more difficult to develop had high intercept values (Figure 10). In terms of the management parameters, it is evident that cumulative manpower is directly proportional to system difficulty while development time is inversely proportional to system difficulty. In other words, system difficulty reflects programming effort and the time required to produce the system. [Ref. 16: 350, 20: 35-36, 22: 20]

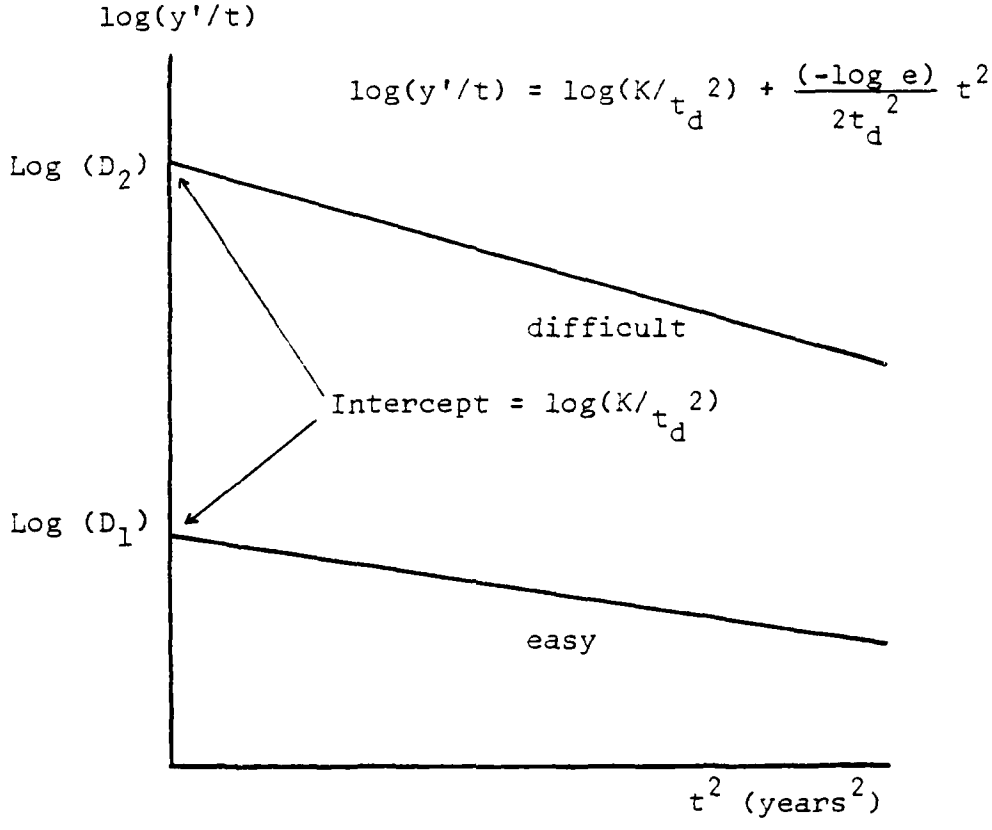


Figure 10

Linear Form of the Rayleigh Equation
[Refs. 16: 350, 20: 36, 22: 227]

Professor George J. Fix, of Carnegie-Mellon University, shows that the dependence of the system difficulty, D, is such that D is a function of time, cumulative manpower, and on-hand manpower.

$$D = D(t,y,y')$$

This implies that difficulty increases with the number of people involved in the activity, both on-hand and cumulative, and time. [Ref. 8: 363]

1. Feasible Region

One can visualize a feasible software region based on development time and life-cycle man years (Figure 11). A system can span from a life-cycle man-year of one to almost any given total number of life-cycle man-years. The development time can range from one month to ten years or more. For large systems, though, bounds must be established, thereby creating a development-feasible region. Development time can be limited from two to five years: five years being an economic upper bound, two years reflecting the lower limit due to limitations on manpower buildup. [Ref. 16: 351, 20: 37, 24: 115]

Frederick Brooks alludes to this lower bound by citing V.A. Vyssotsky's estimates that "a large project can sustain a manpower buildup of 30 percent per year." [Ref. 5: 179] Norden's equation

$$y' = 2Kate^{-at^2}$$

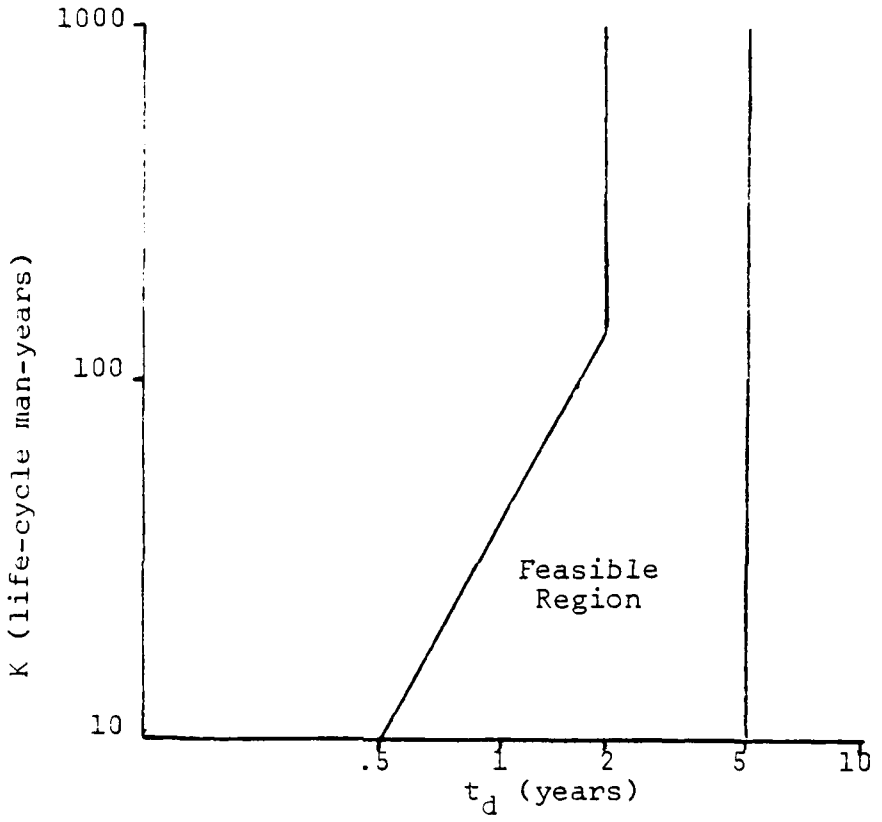


Figure 11

Feasible Software Development Region
[Ref. 20: 37]

holds to this principle when the development time is equal to or greater than two years. [Ref. 22: 23] The manpower buildup invokes Brooks' intercommunication law for those projects where the parts must be separately coordinated with the other parts. This law states that effort increases as a function of n(n-1)/2. By adhering to this law, "three workers require three times as much pairwise intercommunication as two; four require six times as much as two." [Ref. 5: 18] Lawrence Putnam rationalizes this lower limit even more plainly. He says that large software projects less than two years in duration cannot be managed "without heroic measures." [Refs. 16: 351, 22: 24]

By portraying the feasible region in three dimensions through the process of adding a new axis reflecting system difficulty, k/t_d^2, a difficulty surface is created (Figure 12). It should be noted that just as in Figure 10, as development time is decreased, difficulty increases.

2. Difficulty Gradient

Systems tend to fall on lines which correspond to a constant degree of difficulty on the difficulty surface. This constant difficulty can be expressed as

$$K/t_d^3 \sim |\nabla D|.$$

This difficulty gradient reflects the organizational capability while doing that type of work for which the constant

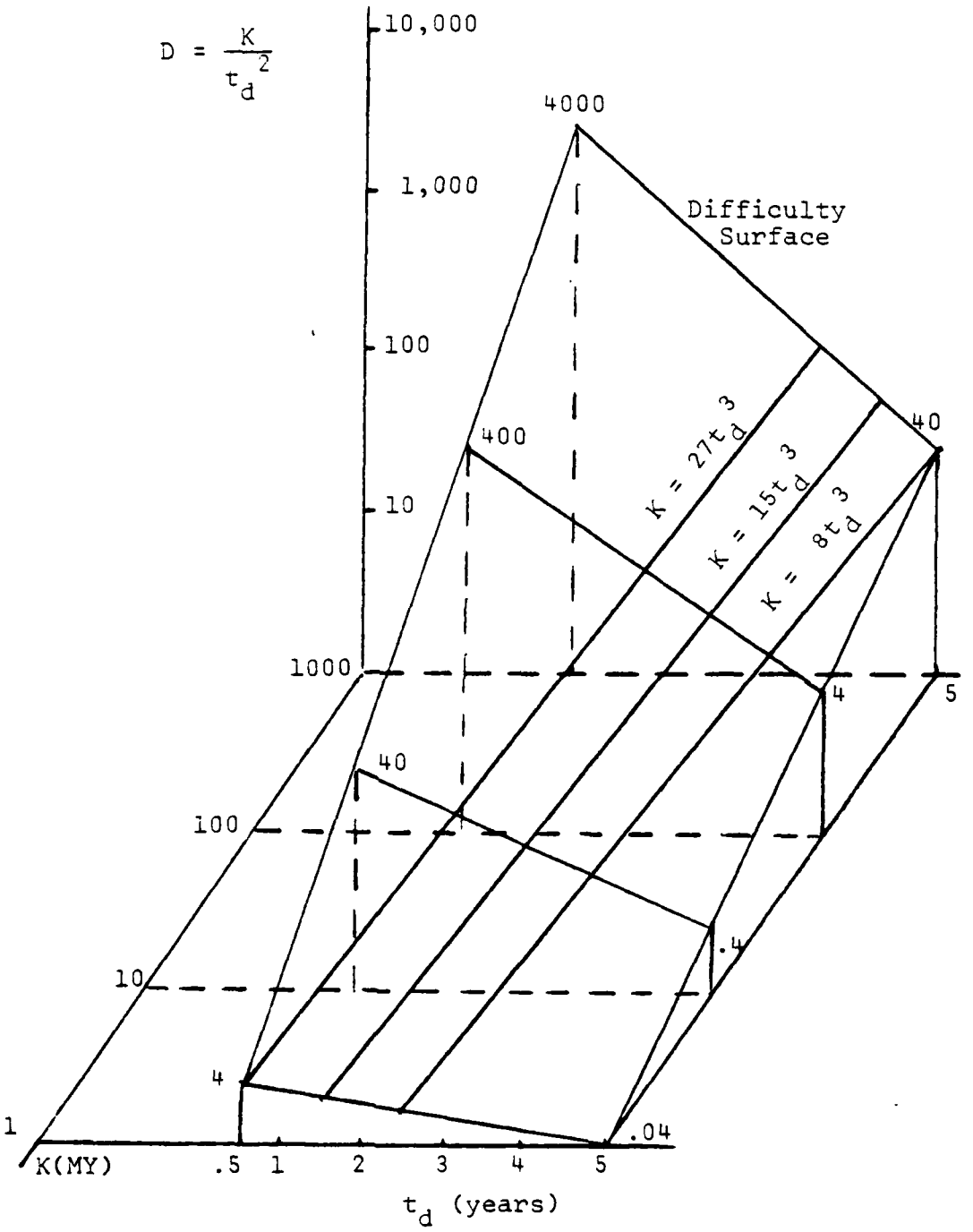


Figure 12

Effort-Time-Difficulty Surface

was derived. [Myers: Ref. 30] As system size increases, development time also increases in order to maintain a constant magnitude of ∇D. [Ref. 15: 352]

Putnam found, through studying all the systems of the U.S. Army's Computer Systems Command, that if a system is entirely new, designed and coded from scratch and containing many interfaces with other systems, $|\nabla D| = 7.3$. If a system is a new stand-alone system, $|\nabla D| = 14.7$. If the system is rebuilt from existing systems where large segments of logic and code exist, $|\nabla D| = 26.9$. Other data from IBM and the Rome Air Development Center has shown that for what Putnam calls a Composite I system, one which consists of independent subsystems which reflect few interactions and interfaces, as well as subsystem development occurring with considerable overlap, $|\nabla D| = 55.0$. For a Composite II system, which is similar to a Composite I but with minimal vice few interactions and interfaces, and with subsystem development occurring essentially in parallel, $|\nabla D| = 89.0$. Putnam does note that as more data becomes available for different classes of systems, more constants are likely to emerge. [Refs. 15: 352, 22: 3, 24: 154]

The values of the difficulty gradient for a particular type of system will vary between software houses. This variance is based on the average skill level of the software house's analysts, programmers, and management. For a particular kind of work, the values of the difficulty gradient

reflect a learning curve for the software house. [Ref. 15: 352] Because of the skill variance and learning attributes, one can expect an uncertainty of fifteen percent for the base gradient values expressed in the previous paragraph. [Ref. 24: 154]

D. PATTERNS OF MANLOADING

There is a myriad of possible manloading patterns. For example, they can be rectangular, trapezoidal, triangular, or just about any geometric shape one can think of. Unfortunately, the perceptual manpower needs of the project may not reflect accurately the real needs. "If management responds promptly to the perceived needs of the ongoing project, the manpower loading pattern will resemble one of the large number of Rayleigh-Norden curves possible." [Ref. 20: 25]

For the example as portrayed in Figure 13, the rectangular manloading pattern, perhaps the simplist for the manager to implement, is shown plotted against a Rayleigh pattern. By applying the concept of the homogeneity of systems development, additional personnel cannot be judiciously applied to the project until the initial work is completed. The rectangular approach results in initial wasted effort, a lack of effort when critically needed, and extra effort toward the end of the project. Because of the lack of effort during the critical periods, schedule

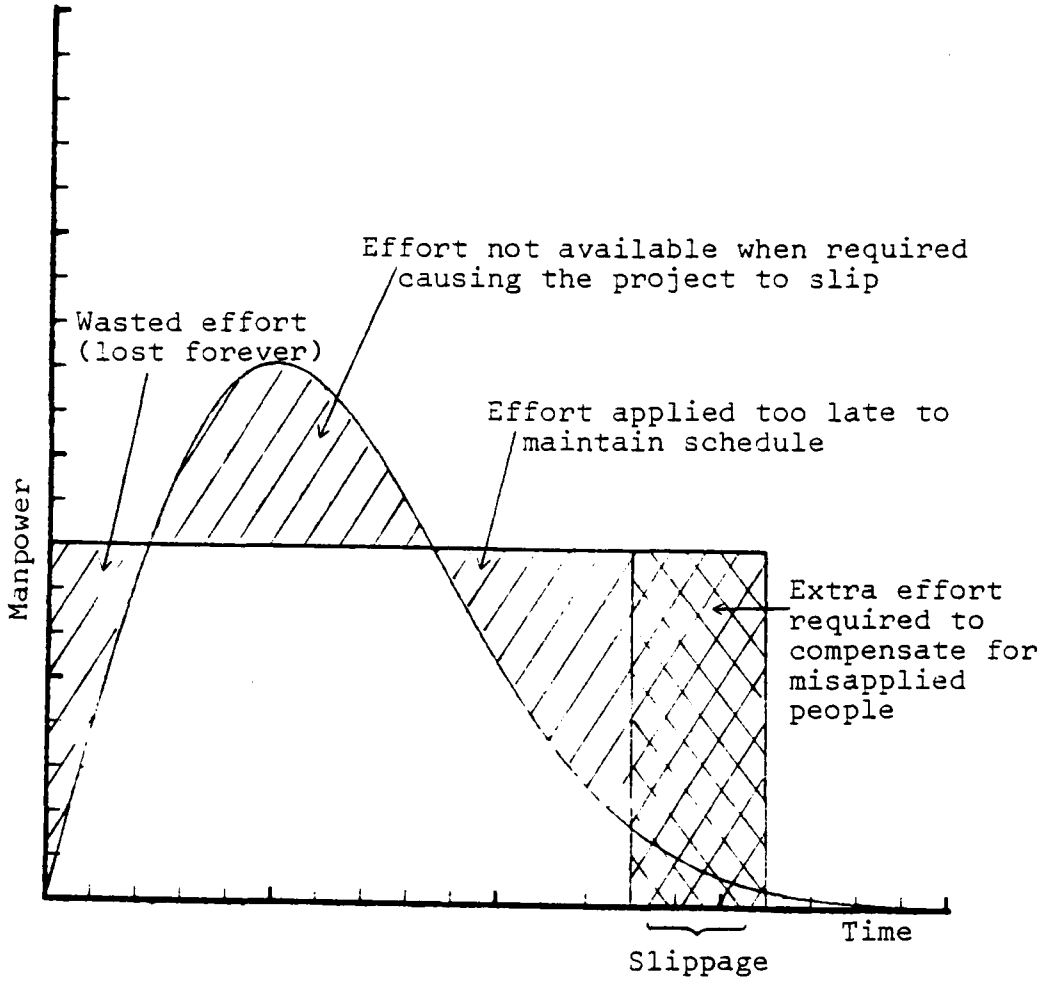


Figure 13

Rectangular Manloading versus Rayleigh Manloading
[Ref. 20: 26]

slippage results and further additional effort is required at the end of the project. This additional effort continues at the maximum manpower level, thus compelling the project to continually impose a maximum cash flow rate and probably forcing a cost overrun. If Brooks' Law is adhered to, this additional manpower will not keep the project on schedule.

Correctly applying manpower involves projecting an expected Rayleigh-Norden curve and setting control limits, based on the uncertainty of the initial data. As more information concerning manpower becomes available, the curve is adjusted to fit reality. The project manager is afforded the luxury of being able to project manpower needs, update his needs, and control the effort.

E. DETERMINATION OF MAJOR MILESTONES

Major milestones are located empirically along the Rayleigh curve based upon the coupling of life-cycle sub-cycles to the overall curve. [Ref. 20: 117] Table 4 shows the milestones derived by Putnam from data from the U.S. Army Computer Systems Command. The figures in Table 4 display quite a range; however, given this range, should a project exceed the maximum time for a particular milestone, it should be quite evident to the project manager that there is a problem with the project that requires attention. The project's not meeting the minimum time should be a signal to the project manager that either he has an exceptional team,

Table 4

Times of Major Milestones
[Refs. 20: 71, 21: 140]

Event	t/t_d
Design Certification	
First	0.235
Expected	0.433
Last	0.618
Systems Integration Test	
First	0.550
Expected	0.660
Last	0.768
Prototype Evaluation Test	
First	0.613
Expected	0.800
Last	0.990
Start Installation/Extension	
First	0.613
Expected	0.930
Last	1.250

NOTE: There is a .90 probability that t/t_d will lie between first and last for a particular milestone.

his planning was not accurate, or something was accomplished in a cursory manner. It has been shown through data from several hundred systems, reflecting various development environments, that the relative occurrence of the four milestones is very stable, with respect to the expected value, in most organizations and environments. [Ref. 21: 140]

It is important to remember that both the development time and the milestones are the most sensitive elements in the system development process. Milestones scale in a linear fashion; thus, if one is late meeting a milestone, he should expect to be late on succeeding milestones. Once behind, the project manager is not afforded the luxury of being able to catch up; "adding manpower to a late software project makes it later." [Ref. 5: 25] Realistic milestones must be set at the beginning of the project. [Ref. 20: 71] Determining the major milestones not only aids the project manager by functioning as a planning tool, but also is a means for measuring actual accomplishment. [Ref. 21: 140]

F. SYSTEM SIZE VERSUS THE LIFE-CYCLE CURVE

The relationship between development effort and life cycle effort

$$E = .4K$$

holds true for large systems, those systems with more than 75,000 source lines of code. This relationship increases in a non-linear fashion as system size decreases.

For small systems, those with less than 18,000 source lines of code, the life-cycle curve closely approximates the curve for the Design and Coding phase (Figure 14). Because there is very little system overhead, the development time, t_d, is near the end of the curve. In small systems, the life cycle curve reaches a peak at

$$t_d/\sqrt{6} \quad \text{[Ref. 20: 76]}$$

The equation for the curve is

$$y' = \frac{K}{t_d^2}\, t\, e^{-3t^2/t_d^2} \quad \text{[Ref. 21: 175]}$$

In the range of 18,000 to 75,000 source lines of code, the overhead support activities (documentation, integration, testing, maintenance, and management activities) rapidly increase. Throughout the range of this transition zone, the life-cycle curve expands from approximating the design and coding curve to the full life-cycle curve.

This range of system size requires an additional parameter in the solution process which makes the solution extremely complicated. The solution involves complex engineering mathematics and is well beyond the scope and intent of this study; however, the solution has been automated and is included in the SLIM (Software Life Cycle Model) software estimation program. SLIM and its functions will be discussed in Chapter V.

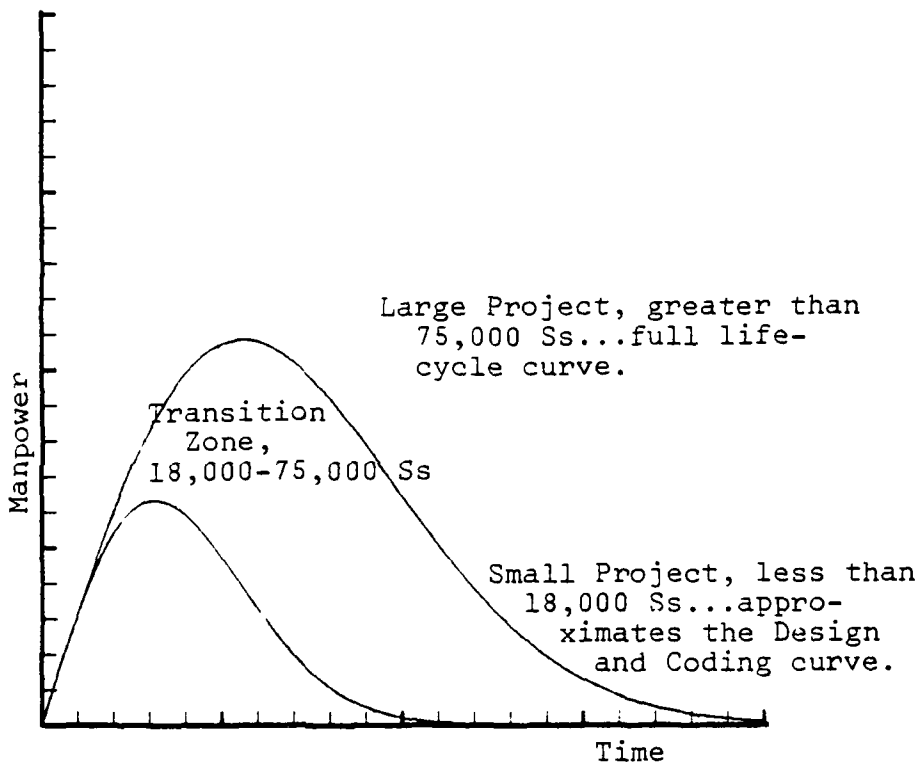


Figure 14

The Life Cycle Curve versus System Size

G. SUMMARY

The curve resulting from the Rayleigh equation

$$y' = \frac{K}{t_d^2}\, t\, e^{-t^2/2t_d^2}$$

can be used to represent the life-cycle of a software project. Even though system noise and random behavior can be expected, time and manpower can be estimated with uncertainties less than three percent and nine percent respectively. Linearizing the Rayleigh equation results in the ability to determine system difficulty. The creation of a three dimensional project feasibility area offers the ability to visualize a difficulty gradient reflecting organizational capability for a particular type of work. Based on empirical data, project milestones can be located along the Rayleigh curve which can, in turn, be valuable as a measure of project control.

IV. SOFTWARE ECONOMICS: THE SOFTWARE EQUATION

In the previous chapter, equations, relationships, and variables were introduced that allow the project manager to plan and control the software development process. Two key variables, life-cycle effort, K, and development time, t_d, afford the project manager the ability to generate both the instantaneous and cumulative manpower requirements, as well as the development costs, through the application of the Rayleigh equation.

This chapter serves to show the relationship of K and t_d to the software development process. This relationship is expressed in a very powerful management tool: the software equation. In addition, the answer to the fourth management question

What are the risks?

will be addressed through the development of risk profiles.

A. THE SOFTWARE EQUATION

The software equation, a mathematical relationship developed by Lawrence Putnam, is an extremely powerful tool for planning and evaluating the development effort of the software life-cycle (Figure 15). The software equation is expressed as:

$$Ss = Ck\ K^{1/3}\ t_d^{4/3}$$

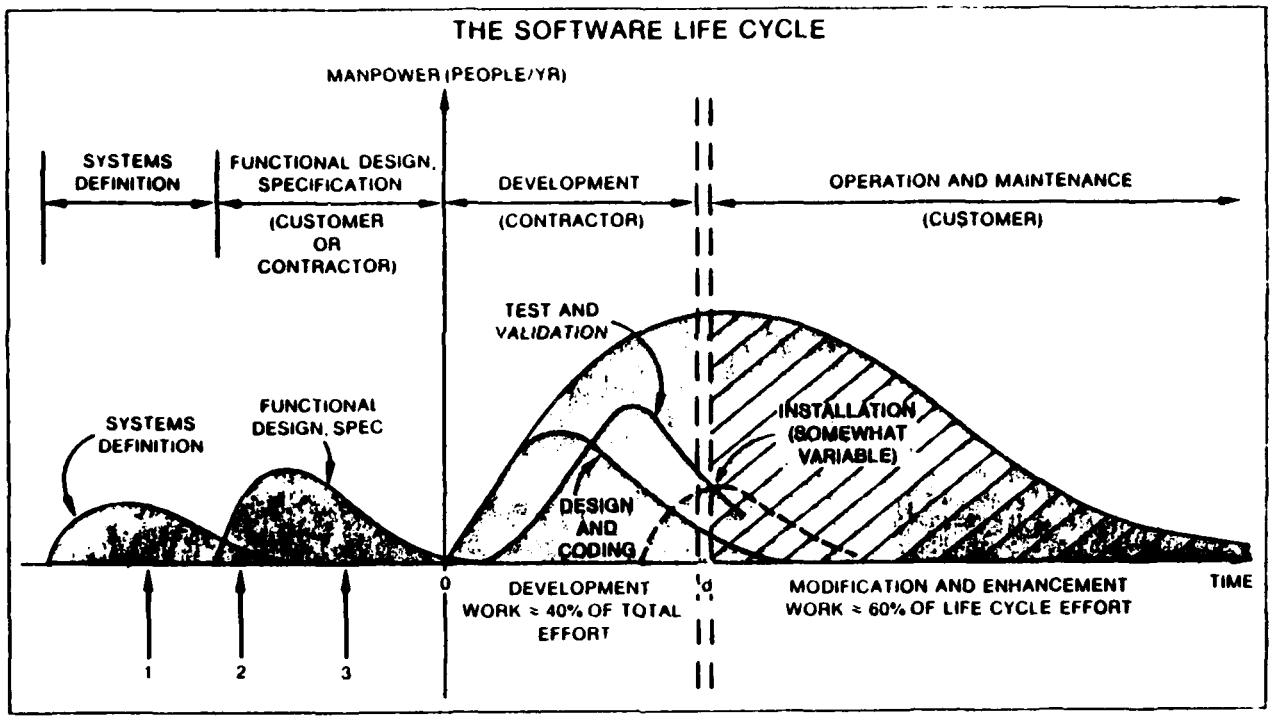


Figure 15

[Ref. 21: 191]

where

Ss = expected number of source lines of code.

Ck = technology constant.

t_d = development time (years).

K = life-cycle effort (man-years).

The mathematical development of the software equation will not be covered in this study: however, an explanation can be found in reference 9: page 828, reference 15: page 353, reference 17: page 352-B and reference 18: page 108.

By rearranging the software equation and applying a factor of .4 to reflect the development effort being approximately the first forty percent of the life cycle curve, the development effort can be determined:

$$E = .4K = .4\left(\frac{Ss}{Ck}\right)^3 \frac{1}{t_d^4}$$

By applying the burdened labor rate, the equation can be converted to a cost function:

$$\text{Development Cost} = (\$/M\text{-}Y)\ .4\ \left(\frac{Ss}{Ck}\right)^3 \frac{1}{t_d^4}$$

[Refs. 19: 804, 20: 7]

1. Technology Constant

The most difficult variable in the software equation to grasp is the technology constant. The technology constant is:

> "the state of technology being applied to the software development; the environment in which the development takes place; the development equipment available, which in turn affects program turnaround and the time needed for debugging and testing; the extent to which modern programming practices are incorporated into the development project." [Ref. 20: 40]

The technology constant for a firm can be calculated given the delivered source lines of code, development effort, and development schedule of completed projects. For a new project, values calculated from other projects completed by the particular software developer can be compared against that which is estimated for the new project to determine if the new constant is reasonable. It is important to remember that a technology constant derived for a particular software house is not necessarily indicative of the technology constant for another software house.

John E. Gaffney, of IBM-Manassas, using data from software development projects performed at IBM, Manassas, Virginia, found a correlation coefficient, r, of -0.72 between the technology constant (Gaffney refers to the technology constant as the development constant) and code complexity. In his study, code complexity is based on the proportion of conditional jumps in the code. This infers

that 51.8 percent of any variation in the technology constant, Ck, can be explained by the measure of code complexity. [Ref. 9: 830]

2. Trade-off Law

The equation for development effort

$$E = .4\left(\frac{Ss}{Ck}\right)^3 \frac{1}{t_d^4}$$

reflects the power of the software equation: the trade-off law.

By improving the development environment, Ck increases, thus decreasing the development effort required to complete a project. By taking out the "whistles and bells" from the system and reducing the number of source statements to, for example, .95 Ss, cost will be reduced to eighty-six percent of the original cost. [Ref. 20: 7] Given a specific environment with source lines of code and technology held constant,

$$K = \frac{\text{Constant}}{t_d^4}$$

effort decreases as the inverse of development time to the fourth power. With a small change in time, a quantum change in effort results (Figure 16). "In software development, time is money. By giving a little (time), we can save a lot (of money)." [Ref. 20: 43]

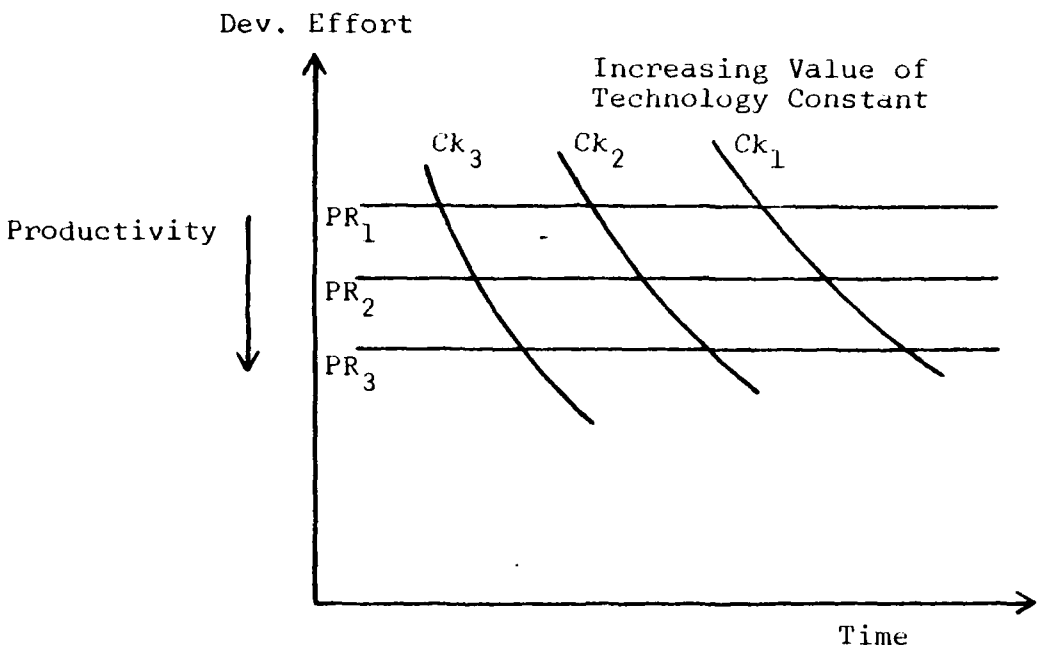


Figure 16

Development Effort, Time, Technology Constant Tradeoff
[Ref. 9: 832]

If one is able to increase his technology constant through, for example, the purchase of a development computer and this increase is such that

$$Ck(new) = 1.3\ Ck(old)$$

the manpower savings can pay for the development computer. Since

$$\text{Development Cost(new)} = \frac{1}{1.3^3}\ \text{Development Cost(old}$$

$$= 45.5\%\ \text{Development Cost(old)}$$

A 54.5 percent savings (100.0 percent - 45.5 percent) on an old development cost of $1 Million would equal $545,000. This savings would certainly be enough to buy a powerful development computer. [Ref. 19: 804-805]

By invoking the tradeoff law, the project manager can effect substantial savings by improving his development environment, keeping the system size as small as practical, and by scheduling as much time as reasonably allowable for the development effort. [Ref. 19: 804]

B. SIZING THE PROJECT

In order to initiate development planning and control, the system size must be determined early in the system definition phase (point 1, Figure 15). Because the actual design functions have yet to be initiated, the project

manager has access only to broad estimates of the system size; yet, this is enough to allow him to determine the basic economic feasibility of the project. There will be a tremendous degree of uncertainty in the early sizing, but this is to be expected. To demand pinpoint accuracy during system definition is an effort in futility.

As the system definition and functional design/specification phases progress, more is known about the system and estimates of system size become more accurate (points 2 and 3, figure 15). By the time that detailed design begins, uncertainty can be reduced to within the limits of engineering accuracy. [Ref. 21: 191]

1. PERT Sizing Technique

Sizing will be performed using PERT (Program Evaluation and Review) sizing techniques. Expected system size is determined by:

$$\text{Expected Size} = \frac{a + 4m + b}{6}$$

where

a = smallest possible system size

m = most likely system size

b = largest possible system size.

This equation is an estimation of the expected value of a Beta distribution and skews the value toward the most uncertain side.

For the first sizing attempt, during the system definition phase, such a wide range of uncertainty exists that a simple average will suffice such that

$$\text{Expected Ss} = \frac{a + b}{2}$$

The standard deviation is determined by the standard deviation of the Beta distribution of system sizes:

$$\sigma Ss = \frac{b - a}{6}$$

This is the range in which 99 percent of the values are expected to occur divided by six. Therefore, the system size will equal

$$Ss = \text{Expected Ss} \pm \sigma Ss.$$

Statistically, there is a fifty percent chance that the system size will fall on either side of the expected system size. There will be a sixty-eight percent certainty of the system being within one standard deviation of the expected system size, ninety-five percent certainty that the system size will be within two standard deviations, and 99 percent certainty that the system size will be within three standard deviations.

The sizing process is demonstrated in the following example.

2. PERT Sizing Example

Early in the system definition phase, the project manager is given the following estimates of system size:

Function	a	m	b
Module 1	76000	90000	117000
Module 2	87000	112000	136000
Module 3	69000	99000	112000

Using the PERT sizing technique, the following expected values and standard deviations are computed:

Function	Expected Ss	σSs
Module 1	92167	6833
Module 2	111833	8167
Module 3	96167	7167
System	300,167	12836

Expected Ss(total) is the summation of the Expected Ss of each module. σSs(total) is equal to

$$\sqrt{\Sigma(\sigma Ss)^2} \ .$$

The results indicate that there is a sixty eight percent certainty that the system size will be

300167 $\pm$ 12836 source lines of code

The numbers are rough estimates and the variance is extremely large; however, the project manager now has reasonable figures within which he can plan.

During the Functional design phase, the project manager receives more detailed estimates based on further modularization of the system.

Function	a	m	b
Module 1A	32075	37175	43900
Module 1B	29550	32429	38475
Module 1C	18225	31582	40900
Module 2A	41875	53450	61248
Module 2B	39950	47120	55353
Module 3A	27210	39805	43715
Module 3B	24900	31625	35775
Module 3C	23875	29375	34345

PERT sizing computations reveal:

Function	Expected Ss	σSs
Module 1A	37446	1971
Module 1B	32957	1488
Module 1C	30909	3779
Module 2A	52821	3229
Module 2B	47297	2567
Module 3A	38378	2751

Module 3B	31196	1813
Module 3C	29287	1745
System	300291	7162

Although the level of detail is still not very great, an even more refined expected number of source statements is now available. What is even more important is that by further modularizing the system, the standard deviation is substantially decreased; in this example, the uncertainty is reduced by more than forty-four percent.

C. DEVELOPMENT TIME/EFFORT DETERMINATION

Far too often, the development time for a project is arbitrarily established by management. Because the development process is so time sensitive,

$$K = \frac{\text{Constant}}{t_d^4}$$

setting development time based on factors external to the project can lead to unanticipated and unwanted difficulties.

For example, the software project in the previous section, in order for it to be ready for a business symposium, has a two year development time imposed upon it. The system is a new stand-alone system with the following parameters:

Ck = 10946
Ss = 300,000
σSs = 5985
$/MY = $50,000

Through use of the software equation, it is calculated that

E = 514.68 man-years

Development Cost ($E) = $25,734,009

This resulting effort and cost is ludicrous.

By plotting the software equation and difficulty gradient as development time versus system size, given a technology constant of 10946 and $|\nabla D| = 14.7$, a tradeoff region is developed (Figure 17). With the difficulty gradient imposing a minimum constraint, it is obvious that the system cannot be developed in two years.

By using the tradeoff region or substituting the difficulty gradient

$$K = |\nabla D| \; t_d^{\,3}$$

into the software equation,

$$t_d = \left(\frac{(\frac{Ss}{Ck})^3}{|\nabla D|} \right)^{1/7}$$

the earliest possible time that the development can be completed is 2.815 years or 33.8 months, almost ten months longer than the time that management arbitrarily set. The system difficulty precludes a development time less than 33.8 months. However, the project can be accomplished such that:

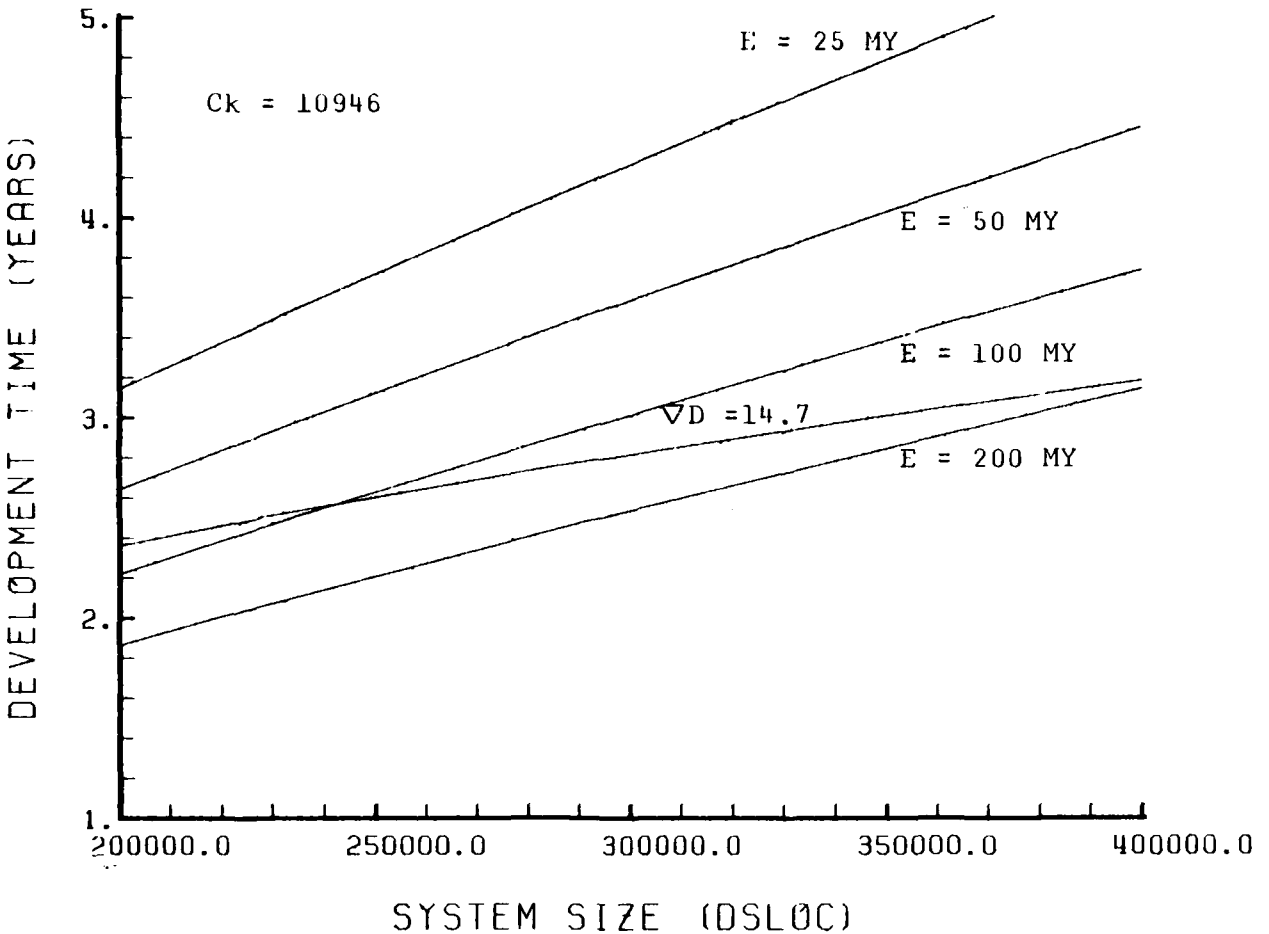


Figure 17

Tradeoff Between Size, Effort and Time

t_d = 33.8 months

K = 327.88 man-years

E = 131.15 man-years

$E = $6,557,723

By increasing the development time by almost ten months, a 74.5 percent reduction in effort and development cost is achieved.

By further increasing development time to three years, more can be saved in terms of effort and development cost:

K = 254.18 man-years

E = 101.67 man-years

$E = $5,083,261

The application of the software equation and the trade-off law provides a powerful basis for a usable software economics relationship. This will allow the project manager to realistically establish time and resource requirements with respect to his development environment.

D. LINEAR PROGRAMMING SOLUTION

Because more than one unknown is being dealt with and management constraints can be applied to the development process, an optimal solution, either in terms of minimum or maximum time, can be reached by using linear programming techniques.

Constraints which can be applied to the linear programming problem are:

Expected source lines of code $Ss = Ck\ K^{1/3} t_d^{4/3}$

maximum peak manpower $K\ t_d \leq \sqrt{e}\ y'\ (max)$

minimum peak manpower $K\ t_d \leq \sqrt{e}\ y'\ (min)$

maximum difficulty gradient $\frac{K}{t_d} \leq 7D$

delivery time $t_d \leq$ contract delivery time

budget $\$/MY(.4K) \leq$ total amt. budgeted for development

with the objective function being the software equation and both K and t_d being the decision variables.

$$Ss = Ck\ K^{1/3}\ t_d^{4/3}$$

Continuing with the previous example, the constraints are:

$|7D| = 14.7$ (Stand-alone system)

$Ss = 300{,}000$

Management has realised the error of their ways and has established a three year development time and an $8 million

development budget. It has also been determined that a minimum of thirty people and a maximum of eighty people will be available at peak manning. Further constraints are:

maximum manpower available	$\leq$	80 people
minimum manpower available	$\geq$	30 people
maximum time	$\leq$	3 years
maximum budget	$\leq$	\$8M

In order to fit the linear programming model, the constraints and objective function must be linearized. By applying logarithms, the objective function and constraints become:

objective function:

minimize

$1/3 \log K + 4/3 \log t_d = \log Ss - \log Ck$

subject to:

$1/3 \log K + 4/3 \log t_d \leq 300000 - \log 10946$

$1/3 \log K + 4/3 \log t_d \geq 300000 - \log 10946$

$\log K - 3 \log t_d \geq \log 14.7$

$\log K - 3 \log t_d \leq \log 14.7$

$\log K - \log t_d \leq \log (\sqrt{e}\ 80)$

$\log K - \log t_d \geq \log (\sqrt{e}\ 30)$

$\log t_d \leq \log 3$

$\log K \leq \log ((8000000/500000)\ (.4))$

Graphically applying the objective function and the constraints results in Figure 18. Because the constraint on the number of source lines of code is graphically parallel to the objective function, there will be an infinite number of solutions with the minimum time/maximum cost and maximum time/minimum cost solutions bounding a feasible region along the source lines of code constraint. In this particular example, t_d and D limit the solution and affect the following optimal solutions:

	t_d(years)	K(MY)	E(MY)	$E
min time	2.82	327.8	131.2	$6,557,723
min cost	3.00	254.2	101.7	$5,083,261

The manpower constraints do not, in this instance, effect the solution, but altering the manpower constraints could result in a totally different optimal solution.

A tradeoff region lies between t_d = 2.82 years/K = 327.8 MY and t_d = 3 years/K = 254.2 MY on the objective function. Clearly, by altering any of the constraints, new optimal solutions can result. Yet, given the constraints of this particular example, the solution is more sensitive to development time as the difficulty gradient imposes the upper bound and cannot be changed.

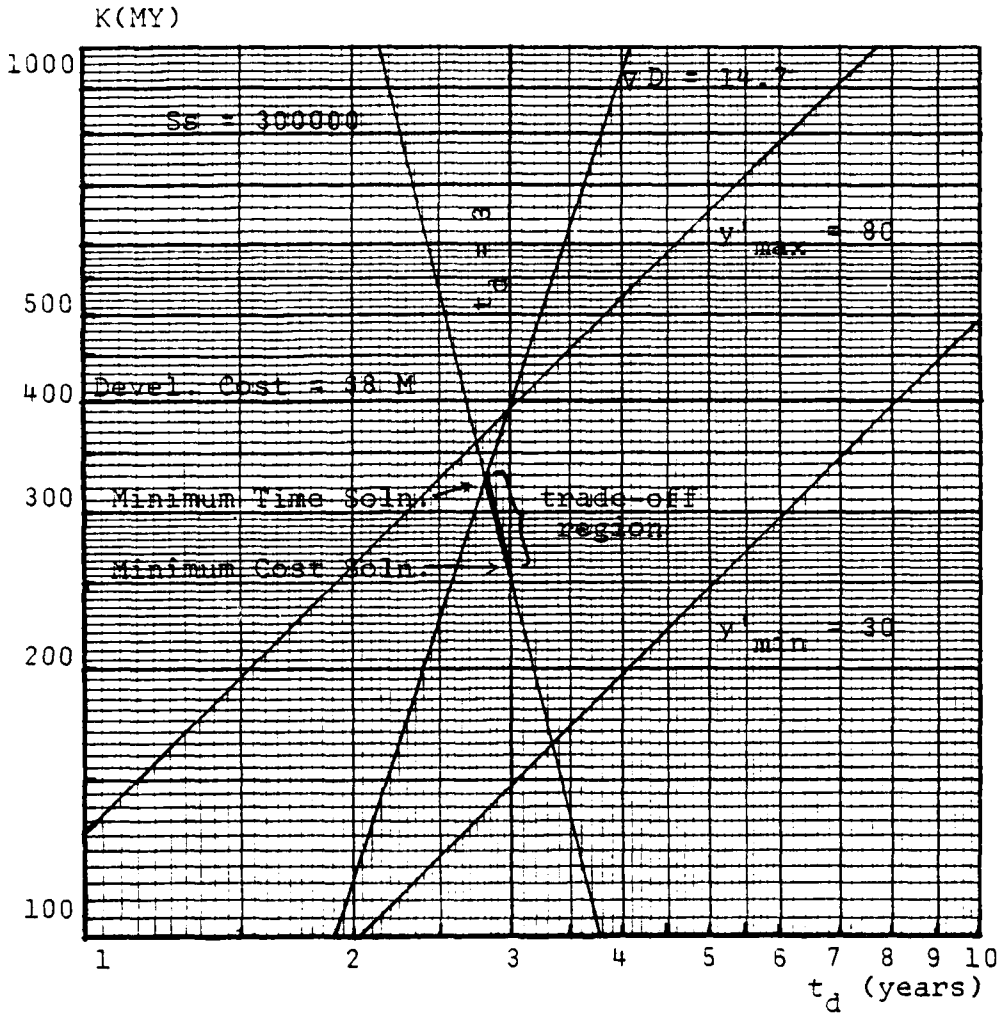


Figure 18

Graphical Linear Programming Solution

E. RISK PROFILES

The life-cycle curve is not a single line, but a line with a bandwidth about it (Figure 9). All the variables of the software equation are subject to some degree of uncertainty and the project manager must have a means of taking this into account in an effort to develop risk profiles.

Already, a degree of uncertainty has been established for the difficulty gradient in Chapter III [Ref. 24: 154] and for source lines of code. Allowing for a standard deviation of fifteen percent of the base value of the difficulty gradient, $|\nabla D| = 14.7$, and for $\sigma Ss = 5985$, as per the previous example, the following equations can be solved similtaneously about the mean, and within $\sigma|\nabla D|$ and σSs:

$$K^{1/3} t_d^{4/3} = \frac{\hat{Ss}}{Ck}$$

$$\frac{K}{t_d^3} = |\nabla D|$$

[Ref. 20: 59]

By solving these equations several thousand times on a computer, $\hat{K}$, $\hat{t_d}$, σK, and σt_d can be determined.

Management has decided to work toward the minimum cost solution

t_d = 3 years

K = 254.18 man-years

E = 101.67 man-years

$E = $5,083,261

and, after performing a simulation, arrives at the following values:

$\hat{t}_d$ = 3 years

σt_d = .25 years

$\hat{K}$ = 252.0 man-years

σK = 20.4 man-years

Risk profiles can now be graphically established for both development time and effort using normal probability graph paper.

Using the hypothetical example, to establish a risk profile for development time, the expected value, 36 months (three years), is plotted at the fifty percent probability level (Figure 19). Below this plot, at the sixteen percent probability level, one standard deviation, 3 months, is marked off. Because the scale of the paper straightens out the normal probability integral, only these two points are required to plot the graph. [Ref. 22: 84]

The risk profile for development time shows a ninety percent confidence interval between 32.2 months and 37.8 months. The project manager can be assured that there is a

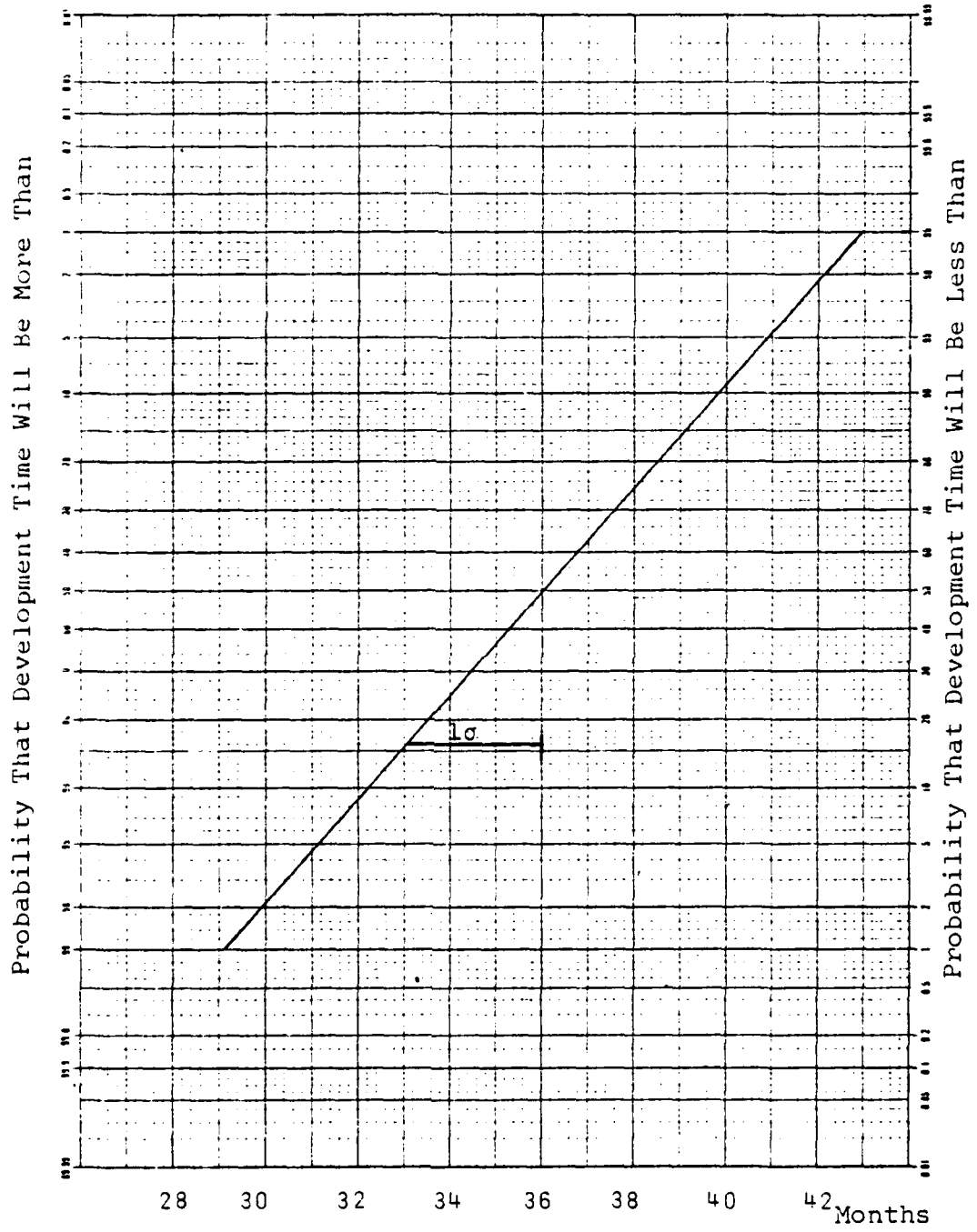


Figure 19

Risk Profile: Development Time

99 percent certainty that the project will not exceed forty-three months, barring external interruptions to the existing development environment.

Likewise, a risk profile can be determined for life-cycle effort, K, (Figure 20) and given an average burdened cost, this can be easily converted to a life-cycle cost risk profile. Also, risk profiles can be developed for development effort and development cost.

F. SUMMARY

The economics of software can be expressed through the mathematical relationship known as the software equation:

$$Ss = Ck\ K^{1/3}\ t_d^{4/3}$$

The software equation is a very powerful managment tool which mathematically reflects important economic and behavioral characteristics to all those concerned with the software project.

The software equation reflects the time sensitivity of the development process. Time cannot be changed without changing effort.

Each software project has an associated minimum development time. Completed development cannot be expected to occur prior to this minimum time. Knowing the minimum development time and its uncertainty gives the project manager a framework from which to plan and control the software development process.

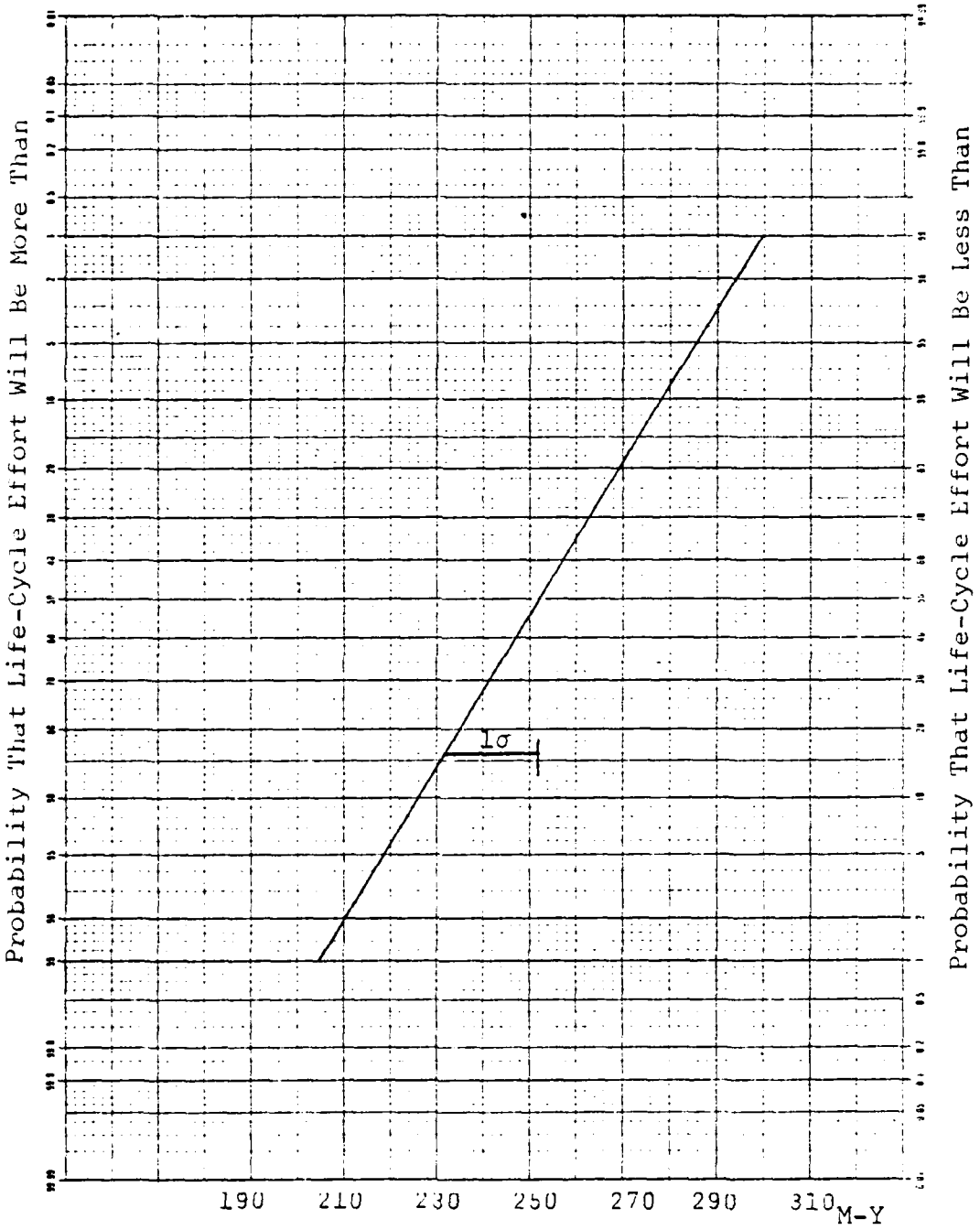


Figure 20

Risk Profile: Life-Cycle Effort

The trade-off law is an extremely powerful tool which offers the project manager the opportunity to save money by allowing more time for the development process, changing the development environment, and eliminating the "whistles and bells" from the project.

V. SLIM: AN AUTOMATED APPROACH

A methodology which permits the project manager to answer the four management questions

Is the project feasible?

What are the resource requirements?

How long will it take?

What are the risks?

has now been mathematically shown. Applying the techniques by hand is, at best, a tedious and time consuming chore. The process has been automated by Quantitative Software Management, Inc., and is available through American Management Systems, Inc. The Department of Defense has taken an interest in this particular methodology and has contracted the services of this automated software estimation package as well as an instructional package as per Government Contract MDA903-81-D-0062 (Appendix B).

This chapter will be devoted to the SLIM (Software Life Cycle Model) package, what it does, its weaknesses, and what it can do for the project manager.

SLIM "is a versatile, highly flexible software system that is designed to help software managers and analysts estimate the cost, manpower, and time to build" software systems. [Ref. 25: 1-1] All outputs are in terms of usable management parameters.

SLIM automates three functions (Table 5). The Estimate function has various options (Table 6) which allow the user to design the system development process, update an ongoing development process, and validate vendor proposals.

A. SLIM EXAMPLE

Perhaps the easiest method to fully explain SLIM is to demonstrate the power and flexibility of the system through an example.

1. Scenario

The software project 'Thesis Example' is a rebuild, business application, system. The project has just entered the functional design phase. The project manager, in order to update development plans, wishes to use SLIM given the following constraints:

Development time: 42 months

Development begins: January 1983

Maximum budgeted cost: $5.5M

Maximum number of men available at peak manning: 60

Minimum number of men available at peak manning: 30

Average burdened labor rate ($/MY): $50,000

Cost of Capital: 10%

In addition, the following is known about the system development environment:

Percentage of on-line development: 100%

Percentage of the development computer dedicated to the development effort: 80%

Table 5

SLIM Functions

Calibrate	Is used to obtain historical data from the user. This data will be translated internally into a "State of Technology" factor for the user's organization. It is then used to calibrate the model to the way a particular organization typically develops software.
Editor	Allows the user to interactively create a data file describing a new software system.
Estimate	Is used to obtain cost, schedule, and manpower estimates once a data file has been built.
Bye	Is used to end the current session.

Table 6

SLIM Options

LINEAR PROGRAM	- This function uses the technique of linear programming to determine the minimum effort (and cost) or the minimum time in which a system can be built. The results are based on the actual manpower, cost, and schedule constraints of the user, combined with the system constraints described earlier to yield a constrained optimal solution.
NEW TIME	- SLIM will automatically determine the minimum time schedule for which development of your system is feasible. This function may be used to set an alternative schedule for development. A new corresponding cost will be provided.
DESIGN TO COST	- This function is used to allow the user to set a new level of effort and cost for development. A new time schedule will be generated.
PERT SIZING	- This technique is used to generate the best estimate of total source statements and the associated standard deviation.
TRADE-OFF ANALYSIS	- This is a very powerful technique which allows the user to compare the costs and manpower necessary to build a system over various time scheudles. This analysis demonstrates the extreme time sensitivity of developing software - as time decreases, the costs go up dramatically. Also, a minimum feasible time schedule is shown, indicating that for the size and type of system is shown, indicating that for the size and type of system described, a shorter time schedule simply cannot be imposed upon it.

Table 6 continued-

FRONT-END ESTIMATES	- This function provides low, average, and high estimates of the time and effort required for the feasibility study and functional design.
MANLOADING	- This function is based on a monte carlo simulation of total manpower and time. Projections of the mean number of people (and the standard deviation) on a month-to-month basis are provided. These projections are based on an optimal application of resources throughout development.
CASHFLOW	- This function is based on a Monte Carlo simulation of manpower, time, $/MY, and inflation rate to provide projections of the expected cashflow on a month-to-month basis throughout development.
LIFE CYCLE	- This function provides a table of expected manpower and cashflow over time throughout the operations and maintenance phase.
RISK ANALYSIS	- This function determines the probability of developing a system within a specified time or for a specified cost. It is very useful for strategic planning purposes, by providing the risk associated with various time & cost decisions.
BENEFIT ANALYSIS	- This function computes the benefit of your system in $/year required to amortize the cost of development and maintenance. It is based on the anticipated economic life of the system as well as the average rate of return for your organization.
MILESTONES	- Based on a predetermined total development time, this function provides a realistic schedule for the major milestones of the project.

Table 6 continued-

CPU USAGE	- This function provides a table of expected machine usage over the life cycle of the system, along with an estimate of total machine hours required for development.
DOCUMENTATION	- Based on the total system size and data from hundreds of similar software systems, a range of expected pages of documentation is given.
ALL ANALYSES	- This function calls all of the above functions automatically except *new time* and *design to cost*. If a complete analysis of a system is desired, it is recommended that you use *new time* or *design to cost* to set your desired schedule, and then call *all analyses* for a complete analysis.
END	- This command may be used to return to the SLIM system monitor.

Percentage of system coded in a high order language (HOL): 100%

HOL used: COBOL

Percentage of the target machine's memory utilized by the system: 50%

Percentage of real-time code: 0.0%

Technology factor that has been calibrated: 9

In addition, modern programming practices will be used extensively except that there will be no chief programmer teams. The project personnel are very experienced in terms of skill and familiarity with COBOL; however, they have only an average experience with the development computer and with a system of this size.

The technology factor was derived using the Calibrate function and historical data. The difference between the technology factor and the technology constant will be discussed later in this chapter.

2. SLIM Application

Given the previous information, the project manager can call on the Editor function to build a file for the project.

```
**********************************************************************
                              EDITOR
**********************************************************************

THE EDITOR ALLOWS THE USER TO INTERACTIVELY SET UP A NEW DATA FILE
DESCRIBING A SOFTWARE DEVELOPMENT SYSTEM.  YOU WILL BE PROMPTED FOR ALL
INFORMATION.  THE RESPONSES YOU PROVIDE WILL BE USED TO DETERMINE:

    (1) THE TYPE AND DIFFICULTY OF YOUR SYSTEM,
    (2) THE STATE OF TECHNOLOGY BEING APPLIED TO DEVELOPMENT, AND
    (3) COSTING INFORMATION ABOUT YOUR ORGANIZATION.

ALL INFORMATION WILL BE SAVED ON FILE FOR LATER UPDATES OR ACCESS.

OUTPUT FILENAME? THESIS

ENTER THE TITLE OF THE SOFTWARE SYSTEM> THESIS EXAMPLE

ENTER START DATE (MMYY) > 0183

ENTER THE FULLY BURDENED LABOR RATE ($ MY) AT YOUR ORGANIZATION> 50000

ENTER THE STANDARD DEVIATION OF YOUR LABOR RATE> 5000

ENTER THE ANTICIPATED INFLATION RATE AS A DECIMAL FRACTION> .1

ENTER THE PROPORTION OF DEVELOPMENT THAT WILL OCCUR IN ONLINE,
INTERACTIVE MODE> 1.0

ENTER THE PROPORTION OF THE DEVELOPMENT COMPUTER THAT IS  DEDICATED
TO THIS SYSTEM DEVELOPMENT EFFORT> .8

ENTER THE PROPORTION OF THE AVAILABLE CAPACITY OF THE DEVELOPMENT COMPUTER
THAT IS USED FOR OTHER PRODUCTION WORK> .2

ENTER THE PROPORTION OF THE SYSTEM THAT WILL BE CODED IN A HOL> 1.0

ENTER THE NUMBER CORRESPONDING TO THE  PRIMARY LANGUAGE TO BE USED:

(1) APL          (4) FORTRAN      (7) ALGOL        (10) ASSEMBLER
(2) PL/I         (5) BASIC        (8) JOVIAL       (11) RPG
(3) COBOL        (6) CMS          (9) PASCAL-ADA   (12) OTHER

ENTER NUMBER> 3

ENTER THE NUMBER CORRESPONDING TO THE TYPE OF YOUR SYSTEM:

(1) REAL TIME OR TIME CRITICAL SYSTEM
(2) OPERATING SYSTEM
(3) COMMAND & CONTROL
(4) BUSINESS APPLICATION
(5) TELECOMMUNICATION / MESSAGE SWITCHING
(6) SCIENTIFIC SYSTEM
(7) PROCESS CONTROL

  4
```

```
CHOOSE THE RESPONSE BELOW WHICH BEST DESCRIBES YOUR SYSTEM.

  (1) THE SYSTEM IS ENTIRELY NEW - DESIGNED AND CODED FROM SCRATCH.  IT HAS
      MANY INTERFACES AND MUST INTERACT WITH OTHER SYSTEMS WITHIN A TOTAL
      MANAGEMENT INFORMATION SYSTEM STRUCTURE.

  (2) THIS IS A NEW STAND-ALONE SYSTEM.  IT IS ALSO DESIGNED AND CODED FROM
      SCRATCH BUT IS SIMPLER BECAUSE THE INTERFACE PROBLEM WITH OTHER SYSTEMS
      IS ELIMINATED.

  (3) THIS IS A REBUILT SYSTEM WHERE LARGE SEGMENTS OF EXISTING LOGIC EXIST.
      THE PRIMARY TASKS ARE RECODING, INTEGRATION, INTERFACING, AND MINOR
      ENHANCEMENTS.

  (4) THIS IS A COMPOSITE SYSTEM MADE UP OF A SET OF INDEPENDENT SUBSYSTEMS
      WITH FEW INTERACTIONS AND INTERFACES AMONG THEM.  DEVELOPMENT OF THE
      INDEPENDENT SUBSYSTEMS WILL OCCUR WITH CONSIDERABLE OVERLAP.

  (5) THIS IS A COMPOSITE SYSTEM MADE UP OF A SET OF INDEPENDENT SUBSYSTEMS
      WITH A MINIMUM OF INTERACTIONS AND INTERFACES AMONG THEM.  DEVELOPMENT
      OF THE INDEPENDENT SUBSYSTEMS WILL OCCUR VIRTUALLY IN PARALLEL.

PAST DATA HAVE SHOWN THAT LARGE SYSTEMS (>200,000 LINES) ARE TYPICALLY OF
TYPE 3, 4, OR 5.

ENTER NUMBER   3

ENTER THE PROPORTION OF MEMORY OF THE TARGET MACHINE THAT WILL BE UTILIZED
BY THE SOFTWARE SYSTEM> .5

ENTER THE PROPORTION OF REAL-TIME CODE> .0

BELOW IS A SET OF MODERN PROGRAMMING TECHNIQUES THAT MAY BE USED ON
A SOFTWARE DEVELOPMENT PROJECT.   DECIDE EACH ARE 3 POSSIBLE
RESPONSES INDICATING THE DEGREE OF USAGE ON YOUR SYSTEM.

     TECHNIQUE                                  RESPONSE

 STRUCTURED PROGRAMMING              (1) <25%   (2) 25-75%   (3) >75%
 DESIGN & CODE INSPECTION            (1) <25%   (2) 25-75%   (3) >75%
 TOP-DOWN DEVELOPMENT                (1) <25%   (2) 25-75%   (3) >75%
 CHIEF PROGRAMMER TEAMS              (1) <25%   (2) 25-75%   (3) >75%

 ENTER 1,2, OR 3 TO INDICATE THE DEGREE OF USAGE EXPECTED ON YOUR SYSTEM.
 ENTER ALL 4 RESPONSES ON 1 LINE, SEPARATED BY A COMMA> 3,3,3,1

 BELOW ARE 4 INDICATORS OF PERSONNEL EXPERIENCE THAT CAN IMPACT THE COST
 AND TIME TO DO A PROJECT.  DECIDE EACH ARE 3 POSSIBLE ANSWERS INDICATING THE
 DEGREE OF EXPERIENCE.

     PERSONNEL EXPERIENCE                       RESPONSE

 -OVERALL SKILL & QUALIFICATIONS     (1)MINIMAL  (2)AVERAGE  (3)EXTENSIVE
 -WITH DEVELOPMENT COMPUTER          (1)MINIMAL  (2)AVERAGE  (3)EXTENSIVE
 -WITH PROGRAMMING LANGUAGE(S)       (1)MINIMAL  (2)AVERAGE  (3)EXTENSIVE
 -WITH SYSTEM OF SIMILAR SIZE
  AND APPLICATION                    (1)MINIMAL  (2)AVERAGE  (3)EXTENSIVE

 ENTER 1, 2, OR 3 TO INDICATE THE DEGREE OF EXPERIENCE OF YOUR PERSONNEL.
 ENTER ALL 4 RESPONSES ON ONE LINE, SEPARATED BY A COMMA> 3,2,3,2

 ENTER YOUR STATE OF TECHNOLOGY FACTOR> 3
```

```
YOU MAY ENTER SIZING INFORMATION IN ONE OF 2 FORMS:
  (1) AN OVERALL RANGE OF SIZE, OR
  (2) RANGES OF SIZE ON A MODULE-BY-MODULE BASIS.
ENTER 1 OR 2 TO INDICATE HOW YOU WANT TO ENTER SIZING DATA> 2


THE FOLLOWING INFORMATION IS REQUIRED FOR EACH OF THE MAJOR
FUNCTIONS IN THE SYSTEM:

   FUNCTION NAME - (UP TO 20 CHARACTERS)
   A  -  THE SMALLEST POSSIBLE NUMBER OF SOURCE STATEMENTS
   M  -  THE MOST LIKELY NUMBER OF SOURCE STATEMENTS
   B  -  THE LARGEST POSSIBLE NUMBER OF SOURCE STATEMENTS

ALL INFORMATION FOR EACH FUNCTION SHOULD BE ENTERED ON 1 LINE,
SEPARATED BY COMMAS.


ENTER THE NUMBER OF MAJOR FUNCTIONS> 8

ENTER FUNCTION NAME, A, M, AND B FOR FUNCTION #  1.
> FUNCTION1,12000,21500,29800

ENTER FUNCTION NAME, A, M, AND B FOR FUNCTION #  2.
> FUNCTION2,16000,27200,31400

ENTER FUNCTION NAME, A, M, AND B FOR FUNCTION #  3.
> FUNCTION3,11300,19000,24200

ENTER FUNCTION NAME, A, M, AND B FOR FUNCTION #  4.
> FUNCTION4,5500,9500,14050

ENTER FUNCTION NAME, A, M, AND B FOR FUNCTION #  5.
> FUNCTION5,8000,13400,19500

ENTER FUNCTION NAME, A, M, AND B FOR FUNCTION #  6.
> FUNCTION6,12500,16650,22500

ENTER FUNCTION NAME, A, M, AND B FOR FUNCTION #  7.
  10000,18200,24700 TO...
FUNCTION7,10000,18200,24700

*** ILLEGAL RESPONSE - PLEASE REENTER> FUNCTION7,10000,18200,24700

ENTER FUNCTION NAME, A, M, AND B FOR FUNCTION #  8.
> FUNCTION8,12500,19500,25500
```

It should be noted that a technology factor, vice a technology constant, is required for SLIM. By using two fibinacci sequences (0,1,1,2,3,5,8... and 0,2,2,4,6,10...), technology constants are internally assigned a technology factor. [Ref. 24: 153] In this example, the technology factor of 9 equates to a technology constant of 5168.

SLIM offers a very powerful tool for firms lacking the historical data necessary to calibrate. A default value of 0 for the technology factor signals SLIM to select a technology factor. This is accomplished by taking the mean technology factor of the Rome Air Development Center and adjusting it in accordance with the answers supplied while using the Editor function.

Now that a file is built for Thesis Example, the project manager would want to use the Estimate function to make pertinent cost, manpower, and time estimates.

```
AVAILABLE FUNCTIONS ARE:
CALIBRATE
EDITOR
ESTIMATE
BYE

FUNCTION? EST

INPUT FILENAME? THEIII

INPUT DATA CHECK - ON

SUMMARY OF INPUT DATA PRINTED (Y OR N)? Y
```

****

SUMMARY OF INPUT PARAMETERS

****

SYSTEM: THEIII EXAMPLE DATE: 12-Sep-81

PROJECT START: 183

COST ELEMENTS			
$/MY	50000.	INFLATION RATE	.100
STD DEV ($ MY)	5000.		
ENVIRONMENT			
ONLINE DEV	1.00	HOL USAGE	1.00
DEVELOPMENT TIME	0.80	PRODUCTION TIME	0.80
LANGUAGE	COBOL		
SYSTEM			
TYPE	BUSINESS APPLICATION	REAL TIME CODE	0.00
LEVEL	3	UTILIZATION	0.50
MODERN PROGRAMMING PRACTICES			
STRUCTURED PROG	3	DESIGN/CODE INSP	3
TOP-DOWN DEVELOPMENT	3	CHIEF PROGRAMMER TEAM	1
EXPERIENCE			
OVERALL	3	SYSTEM TYPE	3
LANGUAGE	3	HARDWARE	3
TECHNOLOGY			
FACTOR	3		

****

SIMULATION

TITLE: THEIL EXAMPLE DATE: 13-SEP-81

*** SIMULATION RUNNING - PLEASE WAIT ***

	MEAN	STD DEV
SYSTEM SIZE (STMTS)	143225.	6265.
MINIMUM DEVELOPMENT TIME (MONTHS)	31.2	0.9
DEVELOPMENT EFFORT (MANMONTHS)	2211.4	227.8
DEVELOPMENT COST (K $1000)		
(UNINFLATED DOLLARS)	9214.	1325.
(INFLATED DOLLARS)	10426.	1514.

SENSITIVITY PROFILE FOR MINIMUM TIME SOLUTION
(EXPECTED VALUES OF TIME, EFFORT, AND COST FOR VARIOUS SYSTEM SIZES)

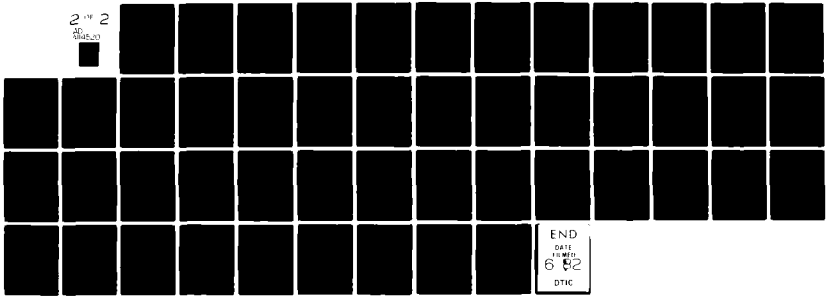
SOURCE STMTS	MONTHS	MANMONTHS	COST (K $1000)
(-3 SD) 124429.	29.3	1851.	7712.
(-1 SD) 136960.	30.5	2092.	8725.
MOST LIKELY 143225.	31.2	2211.	9214.
(+1 SD) 149490.	31.7	2343.	9765.
(+3 SD) 162021.	32.8	2599.	10829.

A CONSISTENCY CHECK WITH DATA FROM OTHER SYSTEMS OF THE SAME SIZE SHOWS:

TOTAL MANMONTHS =	2211.)	GREATER THAN NORMAL EFFORT
PROJECT DURATION =	31.2 MONTHS)	WITHIN NORMAL RANGE
AVG # PEOPLE =	71.)	GREATER THAN NORMAL # OF PEOPLE
PRODUCTIVITY =	65. LINES MM)	LESS THAN NORMAL PRODUCTIVITY)

AD-A114 520 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 9/2
A MACRO APPROACH TO SOFTWARE RESOURCE ESTIMATION AND LIFE CYCLE--ETC(U)
DEC 81 B R VORGANG

UNCLASSIFIED NL



Output is based on PERT sizing and the software equation. The standard deviations are extremely important. Not only do the standard deviations indicate a range about the expected value, but they also offer a device for testing system sensitivity. All simulations are run by generating random variables within the given standard deviations and making use of Monte Carlo simulation techniques.

It should be noted that the consistency check indicates abnormalities within the project that will have to be considered. This consistency check is done against the data base of the Rome Air Development Center.

Given a minimum development time of 31.2 months with a development effort of 2211.4 man-months, the staffing of the development effort, the time of the major milestones, the front-end estimate, and a risk analysis should be examined.

The front-end estimates are based on historical data of system size versus feasibility study time and effort, and system size versus functional design time and effort. Data used for this option comes from IBM-San Jose. The project's development curve is determined and SLIM, in effect, extrapolates backwards to derive the front-end time and manpower requirements. Using system size to estimate the front-end is certainly an unscientific method, hence the large uncertainty bounds. Yet, there is no other feasible means of estimating the front-end of a project but through historical data. Given the uncertainty bounds, this method is quite

satisfactory. Front-end estimation is an important option and is one the project manager would be wise to use; the front-end can incur roughly fifteen to twenty percent of the life-cycle cost.

MANLOADING

TITLE: THEIC EXAMPLE DATE: 12-SEP-81

THE TABLE BELOW SHOWS THE MEAN PROJECTED EFFORT
AND ASSOCIATED +/- OR -/ STANDARD DEVIATION REQUIRED
FOR DEVELOPMENT. THE INPUT PARAMETERS ARE:

	MEAN	STD DEV
DEVELOPMENT EFFORT (MM)	2211.4	227.8
DEVELOPMENT TIME (MONTHS)	31.2	0.9

••• SIMULATION RUNNING - PLEASE WAIT •••

TIME	PEOPLE/MONTH	STD DEV	CUMULATIVE MANMONTHS	CUM STD DEV
JAN 83	3.	0.	3.	0.
FEB 83	9.	1.	12.	1.
MAR 83	14.	2.	26.	3.
APR 83	20.	2.	46.	5.
MAY 83	26.	3.	72.	7.
JUN 83	31.	4.	103.	11.
JUL 83	37.	4.	140.	14.
AUG 83	42.	5.	182.	19.
SEP 83	47.	6.	230.	24.
OCT 83	53.	6.	283.	29.
NOV 83	58.	7.	340.	35.
DEC 83	63.	7.	403.	42.
JAN 84	67.	8.	470.	48.
FEB 84	71.	8.	541.	56.
MAR 84	75.	9.	617.	64.
APR 84	79.	9.	696.	72.
MAY 84	83.	9.	779.	80.
JUN 84	87.	10.	865.	89.
JUL 84	89.	10.	955.	98.
AUG 84	93.	11.	1047.	108.
SEP 84	96.	11.	1143.	118.
OCT 84	99.	11.	1241.	128.
NOV 84	100.	11.	1342.	138.
DEC 84	103.	12.	1444.	149.
JAN 85	104.	11.	1549.	160.
FEB 85	106.	12.	1655.	170.
MAR 85	108.	12.	1762.	181.
APR 85	108.	12.	1870.	193.
MAY 85	108.	12.	1978.	204.
JUN 85	110.	11.	2087.	215.
JUL 85	109.	12.	2197.	226.
AUG 85	55.	6.	2252.	232.

MAJOR MILESTONES

TITLE: THESIS EXAMPLE DATE: 12-Sep-81

EXAMINATION OF SEVERAL HUNDRED SYSTEMS SHOWS THAT ESTIMATES OF THE MAJOR MILESTONES OF A PROJECT ARE VERY STABLE AND PREDICTABLE. THE MILESTONES SHOWN BELOW ARE BASED ON A TOTAL DEVELOPMENT TIME OF 31.2 MONTHS.

EVENT	TIME FROM START (YEARS)	TIME FROM START (MONTHS)
CRITICAL DESIGN REVIEW	1.12	13.4
SYSTEMS INTEGRATION TEST	1.74	20.9
PROTOTYPE TEST	2.08	24.9
START INSTALLATION	2.42	29.0
FULL OPERATIONAL CAPABILITY	2.60	31.2

FRONT-END ESTIMATES

TITLE: THESIS EXAMPLE DATE: 12-Sep-81

	TIME (MONTHS)			EFFORT (MM)		
	(LOW)	(EXPECTED)	(HIGH)	(LOW)	(EXPECTED)	(HIGH)
FEASIBILITY STUDY	7.1	7.8	8.5	8.	31.	55.
FUNCTIONAL DESIGN	9.5	10.4	11.3	199.	398.	597.

RISK ANALYSIS

TITLE: THESIS EXAMPLE DATE: 12-SEP-81

THE TABLES BELOW SHOW THE PROBABILITY THAT IT WILL NOT TAKE MORE THAN THE INDICATED AMOUNT OF TIME, EFFORT, AND DOLLARS TO DEVELOP YOUR SYSTEM.

PROBABILITY	TIME (MONTHS)
1. %	29.1
5. %	29.7
10. %	30.0
20. %	30.4
30. %	30.7
40. %	31.0
50. %	31.2
60. %	31.4
70. %	31.7
80. %	31.9
90. %	32.3
95. %	32.7
99. %	33.3

PROBABILITY PROFILE

PROBABILITY	MANMONTHS	COST (X $1000)	INFLATED COST (X $1000)
1. %	1681.	6130.	6904.
5. %	1837.	7034.	7935.
10. %	1919.	7515.	8486.
20. %	2020.	8099.	9152.
30. %	2092.	8519.	9633.
40. %	2154.	8879.	10043.
50. %	2211.	9214.	10426.
60. %	2269.	9549.	10809.
70. %	2331.	9908.	11219.
80. %	2403.	10329.	11700.
90. %	2503.	10912.	12366.
95. %	2586.	11394.	12916.
99. %	2741.	12297.	13948.

PROBABILITY PROFILE

Knowing that this project definitely cannot exceed 42 months, the project manager can use the Design-to-Risk option to determine what his maximum development time should be in order to assure a 99 percent chance of not exceeding 42 months. If this time is greater than the 31.2 minimum time, the project manager has an opportunity to improve his position by decreasing effort and cost. If the resulting time is less than the minimum time, he can again use this option, but with smaller risk options, in an effort to decrease effort and cost.

```
**********************************************************************
                          DESIGN TO RISK
**********************************************************************
TITLE:  THESIS EXAMPLE                                  DATE: 12-SEP-81


DESIGN-TO-RISK WILL OFTEN IDENTIFY OPPORTUNITIES TO SAVE MONEY USING THE
TRADE-OFF LAW TOGETHER WITH A SPECIFIED LEVEL OF RISK OF NOT EXCEEDING A
REQUIRED DELIVERY DATE.  RESULTS ARE COMPARED TO THE MINIMUM TIME, MAXIMUM
COST SOLUTION.  THE MINIMUM TIME PARAMETERS ARE:

          MINIMUM TIME (MONTHS)                      31.2
          DEVELOPMENT EFFORT (MANMONTHS)           2211.
          DEVELOPMENT COST (K $1000)               9214.



ENTER YOUR MAXIMUM DEVELOPMENT TIME IN MONTHS.
> 42


SPECIFY THE AMOUNT OF RISK ASSOCIATED WITH THE ABOVE SCHEDULE
WHICH YOU ARE WILLING TO ACCEPT:

     (A) VERY HIGH  - 99% PROBABILITY OF NOT EXCEEDING    42.00 MONTHS
     (B) HIGH       - 95% PROBABILITY OF NOT EXCEEDING    42.00 MONTHS
     (C) MEDIUM     - 90% PROBABILITY OF NOT EXCEEDING    42.00 MONTHS

ENTER A, B, OR C
> A


ASSUMING A 99% RISK OF NOT EXCEEDING    42.00 MONTHS, YOUR
EXPECTED (50% LEVEL) PARAMETERS ARE:

                                                    MEAN      STD DEV

          NEW DEVELOPMENT TIME (MONTHS)            39.31        1.14

          NEW DEVELOPMENT EFFORT (MANMONTHS)        872.          30.

          NEW DEVELOPMENT COST (K $1000)           3635.         523.


.....................................................................
. YOUR EXPECTED SAVINGS IN COST BY ACCEPTING A 99% PROBABILITY OF NOT .
. EXCEEDING    42.00 MONTHS IS $    5579. (K $1000) COMPARED WITH THE  .
. MINIMUM TIME SOLUTION.                                             .
.....................................................................


YOUR FILE IS UPDATED WITH THESE NEW PARAMETERS.  RUN MANLOADING AND CASHFLOW
OR LIFE CYCLE TO SEE HOW THESE SAVINGS CAN BE REALIZED.
```

A CONSISTENCY CHECK WITH DATA FROM OTHER SYSTEMS OF THE SAME SIZE SHOWS:

TOTAL MANMONTHS	372.	WITHIN NORMAL RANGE
PROJECT DURATION	39.3 MONTHS	WITHIN NORMAL RANGE
AVG # PEOPLE	22.	WITHIN NORMAL RANGE
PRODUCTIVITY	164. LINES/MM	WITHIN NORMAL RANGE

It should be noted that given a project duration of 39.3 months, the consistency check now shows the project within the normal range of the Rome Air Development Center.

Now knowing that he can plan on a development time of 39.3 months with a 99 percent chance of not exceeding the 42 month limit, the project manager can run the Linear Programming option to determine the time/effort/cost data for a minimum cost solution and a minimum time solution.

LINEAR PROGRAM

TITLE: THEDIS EXAMPLE DATE: 12-SEP-81

THIS FUNCTION USES THE TECHNIQUE OF LINEAR PROGRAMMING (SIMPLEX ALGORITHM) TO DETERMINE THE MINIMUM EFFORT (AND COST) OR THE MINIMUM TIME IN WHICH A SYSTEM CAN BE BUILT. THE RESULTS ARE BASED ON THE ACTUAL MANPOWER, COST, AND SCHEDULE CONSTRAINTS OF THE USER, COMBINED WITH THE SYSTEM CONSTRAINTS YOU HAVE PROVIDED EARLIER TO YIELD A CONSTRAINED OPTIMAL SOLUTION.

ENTER THE MAXIMUM DEVELOPMENT COST IN DOLLARS> 5500000

ENTER MAXIMUM DEVELOPMENT TIME IN MONTHS> 39.31

ENTER THE MINIMUM AND MAXIMUM NUMBER OF PEOPLE YOU CAN HAVE ON BOARD AT PEAK MANLOADING TIME> 30,60

	TIME	EFFORT	COST (X $1000)
MINIMUM COST	39.3 MONTHS	873. MM	3636.
MINIMUM TIME	35.4 MONTHS	1320. MM	5500.

YOUR REALISTIC TRADE-OFF REGION LIES BETWEEN THE LIMITS OF THE TABLE ABOVE.

<INTERPOLATION IN THE TRADE-OFF TABLE BETWEEN THESE LIMITS WILL PRODUCE ALL ACCEPTABLE ALTERNATIVES. WOULD YOU LIKE TO SEE A TRADE-OFF ANALYSIS WITHIN THESE LIMITS <Y OR N> ? Y

TIME	MANMONTHS	COST (X $1000)
35.4	1320.	5500.
36.4	1181.	4921.
37.4	1060.	4416.
38.4	954.	3974.
39.3	873.	3636.

THE RESULTS SHOWN IN THIS TABLE CAN BE USED WITH DESIGN-TO-COST OR NEW TIME TO GENERATE AN UPDATED FILE AND AN ENTIRELY NEW ARRAY OF CONSEQUENT RESULTS FOR MANLOADING, CASHFLOW, LIFE CYCLE, RISK ANALYSIS, COMPUTER TIME AND FRONT END ESTIMATES.

DESIGN TO COST

TITLE: THESIS EXAMPLE DATE: 12-Sep-81

SLIM HAS PROVIDED ITS BEST ESTIMATE OF THE MINIMUM TIME AND CORRESPONDING MAXIMUM EFFORT (AND COST) TO DEVELOP YOUR SYSTEM. THESE VALUES ARE:

MINIMUM TIME:		31.2 MONTHS
EFFORT:		2211. MANMONTHS
COST (W $1000):	$	9214.

A GREATER EFFORT (OR COST) WOULD RESULT IN A VERY RISKY TIME SCHEDULE. HOWEVER, IF A LOWER EFFORT IS SPECIFIED (WITHIN REASONABLE LIMITS), DEVELOPMENT IS STILL FEASIBLE AS LONG AS YOU CAN TAKE MORE TIME.

ENTER DESIRED EFFORT IN MANMONTHS: 873

		MEAN	STD DEV
NEW DEVELOPMENT TIME (MONTHS)		39.3	1.1
NEW DEVELOPMENT COST (W $1000)	$	3638.	206.

YOUR FILE IS UPDATED WITH THESE NEW PARAMETERS. RUN MANLOADING AND CASHFLOW OR LIFE CYCLE TO SEE HOW THESE SAVINGS CAN BE REALIZED.

A CONSISTENCY CHECK WITH DATA FROM OTHER SYSTEMS OF THE SAME SIZE SHOWS:

TOTAL MANMONTHS	873.	WITHIN NORMAL RANGE
PROJECT DURATION	39.3 MONTHS	WITHIN NORMAL RANGE
AVG # PEOPLE	22.	WITHIN NORMAL RANGE
PRODUCTIVITY	164. LINES/MM	WITHIN NORMAL RANGE

Having updated the file for Thesis Example, the project manager can now use the Estimate function to determine the managerial information he requires to plan and control the project.

```
**********************************************************************
                              MANLOADING
**********************************************************************
TITLE:  THESIS EXAMPLE                                  DATE: 12-Sep-81

          THE TABLE BELOW SHOWS THE MEAN PROJECTED EFFORT
          AND ASSOCIATED (+ OR -) STANDARD DEVIATION REQUIRED
          FOR DEVELOPMENT.  THE INPUT PARAMETERS ARE:

                                          MEAN       STD DEV

          DEVELOPMENT EFFORT (MM)        873.0         89.9
          DEVELOPMENT TIME (MONTHS)       39.3          1.1


*** SIMULATION RUNNING - PLEASE WAIT ***
```

TIME	PEOPLE MONTH	ITD DEV	CUMULATIVE MANMONTHS	CUM ITD DEV
JAN 83	1.	0.	1.	0.
FEB 83	2.	0.	3.	0.
MAR 83	4.	0.	6.	1.
APR 83	5.	1.	11.	1.
MAY 83	6.	1.	18.	2.
JUN 83	8.	1.	26.	3.
JUL 83	9.	1.	35.	4.
AUG 83	11.	1.	45.	5.
SEP 83	12.	1.	57.	6.
OCT 83	13.	2.	71.	7.
NOV 83	15.	2.	85.	9.
DEC 83	16.	2.	101.	10.
JAN 84	17.	2.	118.	12.
FEB 84	18.	2.	136.	14.
MAR 84	19.	2.	156.	16.
APR 84	21.	3.	176.	18.
MAY 84	22.	3.	198.	20.
JUN 84	23.	3.	221.	23.
JUL 84	24.	3.	245.	25.
AUG 84	25.	3.	269.	28.
SEP 84	26.	3.	295.	30.
OCT 84	27.	3.	322.	33.
NOV 84	28.	3.	349.	36.
DEC 84	28.	3.	378.	39.
JAN 85	29.	3.	407.	42.
FEB 85	30.	3.	436.	45.
MAR 85	30.	3.	467.	48.
APR 85	31.	3.	497.	51.
MAY 85	32.	3.	529.	54.
JUN 85	32.	4.	561.	58.
JUL 85	33.	4.	593.	61.
AUG 85	33.	4.	626.	64.
SEP 85	33.	4.	659.	68.
OCT 85	34.	4.	693.	71.
NOV 85	34.	4.	727.	75.
DEC 85	34.	4.	760.	78.
JAN 86	34.	4.	795.	82.
FEB 86	34.	4.	829.	85.
MAR 86	34.	4.	863.	89.
APR 86	17.	2.	880.	91.

MAJOR MILESTONES

TITLE: THESIS EXAMPLE DATE: 12-SEP-81

EXAMINATION OF SEVERAL HUNDRED SYSTEMS SHOWS THAT ESTIMATES OF THE MAJOR MILESTONES OF A PROJECT ARE VERY STABLE AND PREDICTABLE. THE MILESTONES SHOWN BELOW ARE BASED ON A TOTAL DEVELOPMENT TIME OF 39.3 MONTHS.

EVENT	TIME FROM START (YEARS)	TIME FROM START (MONTHS)
CRITICAL DESIGN REVIEW	1.41	16.9
SYSTEMS INTEGRATION TEST	2.19	26.3
PROTOTYPE TEST	2.62	31.4
START INSTALLATION	3.05	36.6
FULL OPERATIONAL CAPABILITY	3.28	39.3

CASHFLOW

TITLE: THESIS EXAMPLE DATE: 12-SEP-81

THE TABLE BELOW SHOWS THE MEAN PROJECTED CASHFLOW AND ASSOCIATED STANDARD DEVIATION REQUIRED FOR DEVELOPMENT. THE INPUT PARAMETERS ARE:

	MEAN	STD DEV
DEVELOPMENT EFFORT (MM)	873.	90.
DEVELOPMENT TIME (MONTHS)	39.3	1.1
AVERAGE $/MY (K $1000)	50.	5.
INFLATION RATE	0.100	0.015

*** SIMULATION RUNNING - PLEASE WAIT ***

TIME	$/MONTH (X $1000) MEAN	STD DEV	CUMULATIVE COST (X $1000) MEAN	STD DEV
JAN 83	3.	0.	3.	0.
FEB 83	9.	1.	12.	2.
MAR 83	15.	2.	27.	4.
APR 83	22.	3.	49.	7.
MAY 83	28.	4.	77.	11.
JUN 83	34.	5.	111.	16.
JUL 83	41.	7.	151.	22.
AUG 83	47.	7.	199.	29.
SEP 83	54.	8.	252.	37.
OCT 83	60.	9.	312.	45.
NOV 83	67.	10.	379.	55.
DEC 83	73.	11.	452.	66.
JAN 84	80.	12.	532.	77.
FEB 84	85.	14.	618.	90.
MAR 84	91.	14.	709.	103.
APR 84	98.	16.	806.	117.
MAY 84	103.	16.	910.	132.
JUN 84	110.	17.	1020.	148.
JUL 84	116.	17.	1135.	165.
AUG 84	121.	19.	1257.	183.
SEP 84	126.	20.	1384.	201.
OCT 84	133.	20.	1516.	220.
NOV 84	138.	22.	1654.	240.
DEC 84	144.	23.	1797.	261.
JAN 85	148.	23.	1945.	282.
FEB 85	154.	23.	2098.	305.
MAR 85	158.	24.	2257.	328.
APR 85	161.	23.	2418.	351.
MAY 85	166.	27.	2584.	375.
JUN 85	169.	25.	2753.	400.
JUL 85	174.	27.	2926.	425.
AUG 85	179.	29.	3104.	451.
SEP 85	182.	29.	3286.	477.
OCT 85	185.	28.	3471.	504.
NOV 85	187.	30.	3658.	531.
DEC 85	190.	29.	3848.	559.
JAN 86	192.	31.	4041.	587.
FEB 86	196.	30.	4236.	615.
MAR 86	196.	30.	4433.	644.
APR 86	99.	16.	4581.	544.

THE TABLE ABOVE SHOWS THE AVERAGE $ MONTH AND CUMULATIVE COST IN INFLATED DOLLARS. TO PERFORM THIS ANALYSIS IN CURRENT DOLLARS, CHANGE YOUR VALUE FOR INFLATION RATE TO 0 IN THE DATA FILE FOR THIS SYSTEM.

RISK ANALYSIS

TITLE: THESIS EXAMPLE DATE: 12-SEP-81

THE TABLES BELOW SHOW THE PROBABILITY THAT IT WILL NOT TAKE MORE THAN THE INDICATED AMOUNT OF TIME, EFFORT, AND DOLLARS TO DEVELOP YOUR SYSTEM.

PROBABILITY	TIME (MONTHS)
1. %	36.7
5. %	37.4
10. %	37.8
20. %	38.4
30. %	38.7
40. %	39.0
50. %	39.3
60. %	39.6
70. %	39.9
80. %	40.3
90. %	40.8
95. %	41.2
99. %	42.0

PROBABILITY PROFILE

PROBABILITY	MANMONTHS	COST (K $1000)	INFLATED COST (K $1000)
1. %	664.	3157.	2816.
5. %	725.	3298.	3236.
10. %	758.	3373.	3461.
20. %	797.	3464.	3733.
30. %	826.	3529.	3929.
40. %	850.	3585.	4096.
50. %	873.	3638.	4252.
60. %	896.	3690.	4408.
70. %	920.	3746.	4576.
80. %	949.	3811.	4772.
90. %	988.	3902.	5043.
95. %	1021.	3977.	5268.
99. %	1082.	4118.	5689.

PROBABILITY PROFILE

```
**********************************************************************
                          DOCUMENTATION
**********************************************************************
TITLE:  THESIS EXAMPLE                                  DATE: 12-Sep-91


 IT IS POSSIBLE TO ESTIMATE THE NUMBER OF PAGES OF DOCUMENTATION, BASED
 ON DATA COLLECTED FROM SEVERAL HUNDRED SYSTEMS.

 THE EXPECTED NUMBER FOR YOUR SYSTEM IS      10025 PAGES.

 THE 90% RANGE IS FROM       2864 TO      24348 PAGES.


**********************************************************************



 DO YOU WANT THE PROJECTIONS DISPLAYED  MONTHLY(M), QUARTERLY(Q)
 OR YEARLY(Y)? Q


**********************************************************************
                          LIFE CYCLE
**********************************************************************
SYSTEM:  THESIS EXAMPLE                                 DATE: 12-Sep-91


          THE TABLE BELOW SHOWS THE MEAN PROJECTED EFFORT
          AND CASHFLOW (AND ASSOCIATED STANDARD DEVIATIONS)
          OVER THE LIFE CYCLE OF THE SYSTEM.  ALL
          PROJECTIONS ARE BASED ON AN OPTIMAL APPLICATION OF
          RESOURCES OVER TIME.  THE INPUT PARAMETERS ARE:

                                          MEAN        STD DEV

          DEVELOPMENT TIME (MONTHS)       39.3          1.1
          LIFE CYCLE EFFORT(MM)         3218.7        228.5
          AVG COST/MY (K $1000)           50.           5.
          INFLATION RATE                 0.100         0.015


** SIMULATION RUNNING - PLEASE WAIT **
```

QTR ENDING	PEOPLE		COST/QTR (K $1000)		CUM COST (K $1000)	
	MEAN	STD DEV	MEAN	STD DEV	MEAN	STD DEV
MAR 83	2.	0.	28.	4.	28.	4.
JUN 83	6.	1.	84.	12.	112.	16.
SEP 83	11.	1.	142.	24.	254.	37.
DEC 83	15.	2.	200.	32.	454.	66.
MAR 84	18.	2.	255.	41.	710.	103.
JUN 84	22.	2.	310.	46.	1019.	148.
SEP 84	25.	3.	366.	55.	1383.	201.
DEC 84	27.	3.	416.	64.	1798.	261.
MAR 85	30.	3.	464.	72.	2260.	328.
JUN 85	32.	4.	509.	77.	2766.	402.
SEP 85	33.	4.	535.	86.	3305.	480.
DEC 85	34.	4.	561.	89.	3866.	561.
MAR 86	34.	4.	582.	89.	4448.	646.
JUN 86	34.	4.	595.	93.	5044.	732.
SEP 86	34.	4.	608.	94.	5649.	820.
DEC 86	33.	3.	606.	94.	6256.	908.
MAR 87	32.	3.	604.	94.	6858.	996.
JUN 87	31.	3.	596.	97.	7453.	1082.
SEP 87	29.	3.	587.	93.	8037.	1167.
DEC 87	28.	3.	568.	95.	8604.	1249.
MAR 88	26.	3.	543.	84.	9148.	1328.
JUN 88	24.	3.	512.	84.	9662.	1403.
SEP 88	22.	2.	486.	84.	10147.	1475.
DEC 88	20.	2.	449.	72.	10599.	1539.
MAR 89	18.	2.	427.	73.	11021.	1600.
JUN 89	16.	2.	393.	72.	11415.	1657.
SEP 89	15.	2.	359.	65.	11775.	1710.
DEC 89	13.	2.	329.	62.	12102.	1757.
MAR 90	11.	1.	294.	61.	12398.	1800.
JUN 90	10.	1.	264.	56.	12663.	1839.
SEP 90	9.	1.	232.	49.	12897.	1873.
DEC 90	8.	1.	213.	48.	13106.	1903.
MAR 91	6.	1.	180.	41.	13290.	1930.
JUN 91	5.	1.	162.	39.	13449.	1953.
SEP 91	5.	1.	135.	32.	13587.	1973.
DEC 91	4.	1.	120.	31.	13706.	1990.
MAR 92	3.	1.	101.	27.	13808.	2005.
JUN 92	3.	1.	86.	23.	13895.	2017.
SEP 92	2.	1.	74.	20.	13969.	2028.
DEC 92	2.	0.	62.	19.	14031.	2037.

LIFE CYCLE PROJECTIONS

SLIM is not intended for one-time use. It can serve the project manager throughout the life of the project. As estimates become more refined or as requirements change, the file can be updated and SLIM estimates can serve to aid in the updating of plans and in the control of the project.

B. CONTRACTING APPLICATION

SLIM can be extremely useful in evaluating vendor proposals for software contracts. By requesting the development effort, development time, and system size of past projects, as well as the estimated new project size in PERT format, in the Request for Proposal (RFP), each vendor can be calibrated and the proposals evaluated for validity.

If the user also determines system size, even when using a default technology factor of 0 (zero), a best time/effort/cost estimation can be made from which the vendors can be again be evaluated.

As the number of clients of the SLIM estimating package grows (Table 7), the ability to even further validate vendor proposals also grows. Knowing that a vendor used SLIM to determine his time/effort/cost estimates, the project manager can now verify the proposal using the vendor's SLIM inputs to determine the proposal's validity.

SLIM can do much toward eliminating cost and schedule overruns. Not only can it assist in project development and

Table 7

SLIM Clients as of 20 June 1981

1. American Management Systems, Arlington, Virginia
2. Blue Cross/Blue Shield of Illinois, Chicago, Illinois
3. Boeing Computer Services, Inc., Seattle, Washington
4. Burroughs Corporation, World Headquarters, Detroit, Michigan
5. Burroughs Corporation, Program Products Division, Radnor, Pennsylvania
6. Burroughs Corporation, Program Products Division, Atlanta, Georgia
7. Burroughs Corporation, Program Products Division, Miami, Florida
8. Burroughs Corporation, Program Products Division, Irvine, California
9. Burroughs Corporation, Program Products Division, Charlotte, North Carolina
10. Central Intelligence Agency, Washington, D.C.
11. Computer Management, Inc., Atlanta, Georgia
12. Dynamics Research Corporation, Wilmington, MA
13. Federal Aviation Administration, Washington, D.C.
14. Honeywell Federal Systems Operations, McLean, Virginia
15. Honeywell Large Information Systems Division, Phoenix, Arizona
16. Honeywell Process Management Division, Phoenix, Arizona
17. IBM Federal Systems Division, Manassas, Virginia
18. IBM Federal Systems Division, Westlake Village, California
19. IBM Federal Systems Division, Gaithersburg, Maryland

Table 7 continuted-

20. PACTEL, Ltd., London, United Kingdom

21. Planning Research Corporation, McLean, Virginia

22. United States Department of Defense (includes Army, Navy, Air Force, Marine Corps, and all Defense Agencies)

23. Vought Corporation, Dallas, Texas

Source: Quantative Software Management, Inc.

control, but also by allowing the user to easily determine which vendors are submitting valid proposals.

C. CRITERIA FOR THE GOODNESS OF A SOFTWARE COST MODEL

Some method of evaluating SLIM is necessary to determine its effectiveness as a software costing model. Barry W. Boehm and Ray W. Wolverton, of TRW Defense and Space Systems Group, have proposed nine measures of goodness as a basis on which to compare a software cost model. [Ref. 4: 129]

1. Definition

The model must clearly define which costs it is estimating and which costs it is not.

2. Fidelity

The estimates that are generated by the model must be close to actual expended costs.

3. Objectivity

The model must not allow for subjective factors that can sway the model in any desired direction.

4. Constructiveness

The user must be able to understand why the model gives the estimate it does. Also, the model must help the user understand the software job.

5. Detail

The model must be able to easily subdivide a project into phases and activities.

6. Stability

The model must reflect appropriately sized output per unit of input. No mathematical surprises can be generated by the model.

7. Scope

The model must cover the class of software project to be estimated.

8. Ease of Use

The model's inputs, outputs, and options must be easy to understand and specify.

9. Prospectiveness

The model must not depend on information that is not well known until the end of the project. It must be able to function in a useful manner with the information at hand.

D. EVALUATING SLIM AGAINST THE CRITERIA

Boehm and Wolverton suggest that this criteria is important in determining the utility of a software estimating model. Using this criteria, the utility of SLIM can be assessed.

1. Definition

SLIM clearly defines what it is estimating: size, time, effort, cost, risk, trade-offs, manpower, cashflow, code production, documentation, development computer time, and the front-end of the project.

2. Fidelity

SLIM has been validated against over four hundred projects from the Rome Air Development Center and others.

3. Objectivity

It is extremely difficult to sway SLIM because of the data constraints inherent to the problem as well as those management constraints serving to define the software project's environment. Resultant times less than the minimum possible time are not allowed.

4. Constructiveness

SLIM is completely consistent with the underlying theory. In addition, it offers the user an insight into why constraints are either reasonable or unreasonable. The most important aspect in understanding the software development effort is the identification of minimum time; SLIM immediately identifies minimum time.

5. Detail

The front-end of the project is estimated in terms of time, effort, and manpower. System development is estimated in terms of time, effort, manpower, code production rate, risk, and budget. The Operations and Maintenance phase is estimated in terms of manpower, budget, cost and, risk. Activities are not estimated. Activities are based on specific organization methodology and is within the domain of the project manager, not SLIM.

6. Stability

All output is consistent with the exponential and quantum characteristics of the model.

7. Scope

SLIM is applicable for all classes of software systems involving a group problem solving effort. SLIM is based on the human intercommunication within the software process.

8. Ease of Use

SLIM inputs and outputs are extremely easy to understand. As system design becomes more refined, input ranges become smaller and outputs become more accurate. Output is in terms of usable management parameters.

9. Prospectiveness

SLIM is completely consistent with the amount and availability of information. Based on input characteristics, SLIM's output informs the user of the degree of accuracy.

E. SLIM WEAKNESSES

Although it has been shown to be a very valuable management tool, SLIM does display several weaknesses.

There is a tremendous range of values for the difficulty gradient, $|\nabla D|$, between the rebuild and composite II systems. Although increased uncertainty compensates for this range, further classification of systems within this range of difficulty would further enhance program accuracy.

SLIM can be inconsistent with current Department of Defense life-cycle methodology. SLIM output assumes project continuity. The sequence of events leading to project approval prior to each life-cycle milestone can lead to breaks in the project schedule. This inconsistency, if anticipated by the project manager, must be taken into consideration when planning the software project.

F. SUMMARY

SLIM offers the project manager an extremely powerful tool for managing the software development process in terms of planning, budgeting and control. Sizing through the PERT sizing technique, simulation via the Monte Carlo technique, linear programming, and sensitivity analysis through risk profiles provide an accurate time/effort/cost analysis. SLIM's accuracy has been validated for over four hundred systems spanning the entire range of system types: from operating systems, real-time, and scientific applications to the most mundane business application. SLIM can be used throughout the entire planning and development phases of the project in order to facilitate planning and control. Project data is easily updated so that SLIM can be an effective tool when requirements or constraints change.

Applying SLIM to Barry W. Boehm and Ray W. Wolverton's criteria for the goodness of a software cost model shows SLIM to be an extremely viable software estimation model.

SLIM does display a few minor weaknesses. More system types need to be identified in the difficulty range between rebuild and composite II systems. SLIM output can be inconsistent with current Department of Defense life-cycle methodology. These weaknesses are minor and certainly do not distract from the effectiveness of SLIM as a powerful tool for the software project manager (Table 8).

An extremely important facet of SLIM is that it takes into account the transition zone, as discussed in Chapter III. It is an effective tool for small, medium and large projects. If any two of the following four attributes apply to the project, SLIM can be used:

Ss is greater than or equal to 5000

Peak manpower is greater than or equal to 3 people

Development time is greater than or equal to 6 months

Cost is greater than or equal to $100,000.

Table 8

Putnam's Version of the Characteristics
of a Good Software Cost Estimating System

1. Should have a sound phenomenological basis that relates to other similar, known processes and which contributes to understanding the software system development process.

2. Should apply to all organizations and classes of software.

3. Can be adapted to any organizational structure and way of doing business.

4. Is consistent with existing, known data over the entire size range.

5. Can be easily calibrated or "tuned" to a specific organization.

6. Accepts management constraints on time, cost, manpower.

7. Should have "what if" capability to specify alternative cost, schedule, manpower and risk conditions within constraint bounds.

8. Produces bounded solution sets.

9. Produces accuracy bounds on all answers.

10. Permits risk evaluation and risk specification.

11. Should produce consistent manpower and budget implementation plans for each time-cost-effort solution.

12. Should be capable of adapting to anticipated changes in environment, technology, language, skill, complexity and modern programming practices.

13. Input information is easy to estimate.

14. Gives a measurable achievement projection (rate of code production).

15. Produces a projection of life cycle time, effort, manpower and budget together with accuracy bounds on these quantities.

Table 8 continued-

16. Is capable of adaption to future (new) environments. Makes extrapolation into such new environments straight forward and relatively safe.

17. Allows partitioning into major system phases.

18. Permits separate analysis of modules. Aggregation of these pieces should show when and how much "overhead work" (management, integration, test & validation, documentation) has to be done.

reproduced this with permission of L.H. Putnam, Quantitative Software Management, Inc.

VI. CONCLUSIONS

Planning and controlling the software development process has shown, in the past, to be an extremely difficult task. The estimation of resource requirements, development costs, risk profiles and project feasibility has often proven to be inaccurate, thus costing the user both time and dollars. However, by using obtainable management parameters, and simple engineering and operations research techniques, estimating can be done easily and accurately by taking a macro approach to the estimation problem.

The methodology discussed in the study, as developed by Lawrence H. Putnam, is based on the empirical findings of the relationship between the software life-cycle and the Rayleigh curve, mathematically expressed as

$$y' = 2Kate^{-at^2}$$

The Rayleigh equation can be used to determine project manloading, cumulative manpower, and cash flows.

A powerful software economics tool, the software equation,

$$Ss = Ck\ K^{1/3}\ t_d^{4/3}$$

offers the opportunity to trade-off manpower, development time, and the development environment in order to obtain an optimal development solution in terms of minimum cost or minimum time.

The automated software estimation package, SLIM, makes use of these two equations, as well as engineering and operations research techniques, to form an extremely powerful tool for the project manager.

SLIM and its underlying mathematical basis reflect intuitive observations of the software development process:

Each software system has its own minimum development time

Large projects take more effort

Complex projects require more effort and time

Improving software development tools and techniques results in an improved development environment which, in turn, effects development in a positive manner

A gradual growth in manpower is the most efficient manloading pattern

SLIM is based on the historical data of past development projects which allows it to be calibrated to the development history of the user. In addition, it can function as a 'what if' analysis tool so that the user can design the software development project based on cost, schedule, or risk. SLIM, being completely consistent with the criteria set forth by Barry W. Boehm and Ray W. Wolverton, offers the project manager of any software project a viable software estimation model.

SLIM does, however, reflect several weaknesses: the necessity of identifying further system types in the difficulty range between the rebuild and composite II systems, and the possible inconsistency with current Department of

Defense life-cycle methodology. As previously stated, these weaknesses do not distract from the effectiveness of SLIM as a powerful tool for the software project manager.

By using SLIM, the project manager is able to accurately answer the four management questions in terms of costs and resources:

Is the project feasible?

What are the resource requirements?

How long will it take?

What are the risks?

Being able to answer these questions, the project manager can now prevent the mistakes of past projects and accurately plan and control the software project.

APPENDIX A

Variables That Correlate Significantly With Programming Productivity [Ref. 28: 64, 65]

Question or Variable	Response Group Mean Productivity (DSL/MM)			Productivity Change (DSL/MM)
Customer interface complexity	<Normal 500	Normal 295	>Normal 124	376
User participation in the definition of requirements	None 491	Some 267	Much 205	286
Customer originated program design changes	Few 297		Many 196	101
Customer experience with the application area of the project	None 318	Some 340	Much 206	112
Overall personnel experience and qualifications	Low 132	Average 257	High 410	278
Percentage of programmers doing development who participated in design of functional specifications	<25% 153	25-50% 242	>50% 391	238
Previous experience with operational computer	Minimal 146	Average 270	Extensive 312	166
Previous experience with programming	Minimal 122	Average 225	Extensive 385	263
Previous experience with application of similar or greater size and complexity	Minimal 146	Average 221	Extensive 410	264
Ratio of average staff size to duration(people/month)	<0.5 305	0.5-0.9 310	>0.9 173	132

Hardware under concurrent development	No 297		Yes 177	120
Development computer access, open under	0% 226	1-25% 274	>25% 357	131
Development computer access, closed	0-10% 303	11-85% 251	>85% 170	133
Classified Security environment for computer and 25% of programs and data	No 289		Yes 156	133
Structured programming	0-33% 169	34-66%	>66% 301	132
Design and code inspections	0-33% 220	34-66% 300	>66% 339	119
Top down development	0-33% 196	34-66% 237	>66% 321	125
Chief programmer team usage	0-33% 219	34-66%	>66% 408	189
Overall complexity of code developed	<Average 314		>Average 185	129
Complexity of application processing	<Average 349	Average 345	>Average 168	181
Complexity of program	<Average 289	Average 299	>Average 209	80
Overall constraints on program design	Minimal 293	Average 286	Severe 166	107
Program design constraints on main storage	Minimal 391	Average 277	Severe 193	198
Program design constraints on timing	Minimal 303	Average 317	Severe 171	132
Code for real-time or interactive operation, or executing under severe timing constraint	<10% 279	10-40% 337	>40% 203	76

Percentage of code for delivery	0-90% 159	91-99% 327	100% 265	106
Code classified as non-mathematical application and I/O formatting programs	0-33% 188	33-66% 311	>66% 267	79
Number of classes of items in the data base per 1000 lines of code	0-15 334	16-80 243	>80 193	141
Number of pages of delivered documentation per 1000 lines of delivered code	0-32 320	33-88 252	>88 195	125

APPENDIX B

Department of Defense Memorandum Concerning SLIM

The following memorandum serves to focus Department of Defense interest in SLIM, notify automated data processing users of its implications as a tentative standard Department of Defense methodology, and to notify users of the contracted training sessions conducted by Quantative Software Management, Inc.

OFFICE OF THE ASSISTANT SECRETARY OF DEFENSE

WASHINGTON D C 20301

COMPTROLLER

8 MAY 1981

MEMORANDUM FOR ADP POLICY COMMITTEE (PROGRAM MANAGEMENT)

SUBJECT: Contract Support for Standard Software Development Estimates

We have been working together for some time to improve the Department's capability for estimating software development resource requirements. In 1976 a DoD working group selected a two part methodology. This approach has been under test at a large number of DoD software development centers since 1978.

Recently, the first part of the tentative standard DoD methodology entered the commercial market as an automated model accessible through commercial timesharing. I have become sufficiently impressed with the ease of use and the ready response to "what if" analysis, in the commerical version, to obtain funding for a DoD-wide license for this product. It is my hope that this will expedite widespread use of this capability.

Under the contract which we have negotiated, OSD has paid the annual license fee for DoD-wide use and for four one week training sessions which we plan to hold at DODCI. Any organization in DoD which wishes to use the capability will write a delivery order against our contract and will pay for usage; training for a limited number of persons will be free (except for TDY costs if necessary). Additional details are attached.

It has been demonstrated to my satisfaction that we can save large amounts of money if:

- We do not try to do the impossible, i.e., develop software in less than the minimum time required.

- We make informed decisions about reducing gold plating, i.e., focus software development on the minimum essential requirements.

- We optimize the application of manpower, i.e., a small stretch-out in schedule planned at the outset may substantially reduce overall cost.

- We monitor valid thresholds for deviation from plan, i.e., know what reasonable deviations from plan are.

At least one person at any activity which plans to use SLIM should be trained in the use of the model prior to making any extensive use. The first training session has been filled. We are taking nominations for future sessions. If you wish to nominate people to receive training, please provide my office with the name, organization and telephone number of those persons wishing to attend the 10-14 August session by 30 June. Similar information should be provided by 14 August for the two remaining sessions which have been scheduled for the October-November 1981 and February-March 1982 timeframes. Specific dates on these sessions will be provided later.

Your assistance in giving this information the broadest possible distribution would be appreciated. My points of contact for this effort are Mr. Robert Cooper, 695-2554, (AV 225-2554) and Ms. Scarlett Curry, 697-8632 (AV 227-8632).

John M. Carabello
Chairman

GENERAL CONTRACT INFORMATION

The contract has been issued by:

Defense Supply Service - Washington
Room 1D245, The Pentagon
Washington, D.C. 20310
Attn: Mrs. Katie E. Moulder
AUTOVON 227-6021
Commercial 202 697-6021

The Contract Number is: MDA903-81-D-0062

The contract is for the acquisition of an annual lease and license fee for unrestricted use of a proprietary software package, Software Life Cycle Management Model (SLIM) for all DoD organizations; and for access to the model thru teleprocessing services, also provided for under the contract. The software license fee has been paid by the Office of the Secretary of Defense. SLIM is resident on the American Management Services, Inc., computer systems. Teleprocessing services to access SLIM will be ordered via call orders to the contract.

Complete copies of the contract and additional information may be obtained from the Contracting Officer's Technical Representative:

Rob Cooper
Assistant Director Data Automation
OASD(C) MS DDA Fm 1A658 Pentagon
Washington, D.C. 20301

ACQUISITION APPROVAL

The Office of the Secretary of Defense has approved the acquisition of this capability for software development organizations, organizations that monitor contract software development, and audit or cost estimating organizations.

A delegation of procurement authority has been obtained which covers all DoD activities of the type described above. A "2068" has been processed thru GSA to obtain Sharing Program approval. Any further action of this type will be taken by the COTR if it appears necessary as a result of unexpected growth in DoD wide usage of the contract.

By virtue of the actions described above, potential users within DoD have the authority to issue call orders through their own acquisition activity for use of SLIM, subject to local fund availability.

Call orders which exceed $10,000 should be coordinated telephonically with the COTR prior to issuance to facilitate planning.

ORDERING INFORMATION

Orders may be issued under this contract from 1 May 1981 thru 30 April 1982. Information concerning renewal of the contract will be furnished at a later date.

Orders should be issued thru properly executed DD Form 1155 and mailed to:

American Management Systems, Inc.
1777 Kent Street
Arlington, VA 22209

Two copies of each call order will be sent to the COTR. An Installation Representative (IR) will be designated by each ordering activity. The ordering activity should provide Defense Supply Service Washington with the name, mailing address, and telephone number of the IR. The name and address of the IR will also be cited in the call order* so that invoices may be forwarded to the IR for certification of receipt of services. The call order shall site the cognizant paying office. IR's will certify and approve invoices for payment and forward them to the paying office.

*Block 14 DD Form 1155

PRICING INFORMATION SUMMARY

(These are GSA/TSP prices. Reference should be made to the contract for more detailed information)

Remote Terminal Connect Charges

	Prime Hours	Non-Prime
Washington, DC Metro Area 1200 baud	$7.00/hr	$4.00/hr
Outside Washington, DC up to 1200 baud	$7.00/hr Plus communications charge	$4.00/hr
Communications Charges TELENET	$7.00/hr	$7.00/hr
CPU Utilization Charges* DECSYSTEM-20	$.14/RU**	$.08/RU

* A 50% surcharge will be added to the CPU costs.
** RU = Resource Unit

File Storage Charges (Permanent)	1-1000 pages	$ .64/page/mo
	1001-2000 pages	$ .39/page/mo
	2001-4000 pages	$ .26/page/mo
	Over 4000 pages	$ .18/page/mo

ESTIMATING CALL ORDER COSTS

A rudimentary method for estimating the cost of using SLIM is to estimate the number of manhours available to operate the model, e.g., 2 people X half a day X two days per week equates to 16 hours per week, assuming the people don't work together on one terminal. Costs per hour can be expected to range between $15 and $75 per hour. The variance between these amounts relate to the sophistication of the users and the amount of detailed printing of schedules, etc. Costs at the high end may indicate that a greater benefit is being obtained.

TRAINING INFORMATION

Instructor: Mr. Lawrence Putnam

Training Vendor*:

Quantitative Software Management (QSM)
1057 Waverly Way
McLean, Virginia 22107
(703) 790-0050

*The training vendor will certify attendance and course completion for administration purposes not for billing - training organizations are not billed, as the courses have been prepaid by OSD

Location of Training Site:

DoD Computer Institute
Building 175
Washington Navy Yard
Washington, D.C. 20374

Length of Course: Five (5) days

Course Cost: Tuition - paid by OSD, no cost to trainees' organization

TDY - paid by trainees' organization as required

The following documentation shall be furnished to each trainee:

SLIM Users Guide

AMShare Users Guide

Tutorial Test, "Software Cost Estimating and Life Cycle Control: Getting the Software Numbers," IEEE Cat No EHO 165-1

Booklet, "SLIM" - Sample Output

Folder, QSM, Schematic Diagram of Software Life Cycle Methodology

Student workbook containing exercises and examples in use of SLIM and AMShare

LIST OF REFERENCES

1. Aron, J.D., "Estimating Resources for Large Programming Systems", Software Cost Estimating and Life Cycle Control: Getting the Management Numbers, by Lawrence H. Putnam, The Institute of Electrical and Electronics Engineers, Inc., New York, New York, 1980, pp. 226-237.

2. Biggs, Charles L.; Birks, Evan G.; Atkins, William, Managing the Systems Development Process, Touche Ross and Company, 1980.

3. Boehm, Barry W., "Software Engineering", IEEE Transactions on Computers, Volume C-25, Number 12, December 1976, pp. 1226-1241.

4. Boehm, Barry W.; Wolverton, Ray W., "Software Cost Modeling: Some Lessons Learned", Second Software Life Cycle Management Workshop, The Institute of Electrical and Electronics Engineers, Inc., New York, New York, 1978, pp. 129-132.

5. Brooks, Fredrick, P., Jr., The Mythical Man Month, Addition-Wesley Publishing Company, Reading, Massachusetts, 1975.

6. Department of Defense Directive Number 7920.1 dated 17 October 1978, Life Cycle Management of Automated Information Systems (AIS).

7. Doty, D.L.; Nelson, P.J.; Stewart, Kenneth R., Software Cost Estimation Study, Volume II: Guidelines For Improved Software Cost Estimating, RADC-TR-77-220-Volume II, Doty Associates, Inc., for Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base, New York, August 1979.

8. Fix, George J., "The Dynamics of Software Development I", Software Phenomenology, Working Papers of the Software Life Cycle Management Workshop, U.S. Army Institute for Research in Management Information and Computer Science, Computer Systems Command, U.S. Army, 21-23 August 1977, pp. 362-370.

9. Gaffney, John E., Jr., "A Macro Analysis Methodology for Assessment of Software Development Costs", Symposium on the Economics of Information Processing, IBM Systems Research Institute, 15-19 December 1980, pp. 819-836.

10. Marine Corps Order P5231.1, no date, unpublished draft copy, Volume II: Life Cycle Management for Large Systems.

11. Myers, Ware, "A Statistical Approach to Scheduling Software Development", Computer, Volume 11, Number 12, December 1978, pp. 23-35.

12. Norden, Peter V., "Project Life Cycle Modelling: Background and Application of the Life Cycle Curves", Software Phenomenology, Working Papers of the Software Life Cycle Management Workshop, U.S. Army Institute for Research in Management Information and Computer Science, Computer Systems Command, U.S. Army, 21-23 August 1977, pp. 217-306.

13. Norden, Peter V., "Resource Usage and Network Planning Techniques", Operations Research in Research and Development, edited by Burton Dean, John Wiley & Sons, Inc., New York, 1963, pp. 149-169.

14. "Plenary Session Summary", Software Phenomenology Working Papers of the Software Life Cycle Management Workshop, U.S. Army Institute for Research in Management Information and Computer Science, Computer Systems Command, U.S. Army, 21-23 August 1977, pp. 9-28.

15. Putnam, Lawrence H., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem", IEEE Transactions on Software Engineering, Volume SE-4, Number 4, July 1978, pp. 345-361.

16. Putnam, Lawrence H., "The Influence of the Time-Difficulty Factor in Large Scale Software Development", Digest of Papers, COMPCON 77, Fall, The Institute of Electrical and Electronics Engineers, Inc., New York, New York, 1977, pp. 348-353.

17. Putnam, Lawrence H., "Measurement Data To Support Sizing, Estimating and Control of the Software Life Cycle", Digest of Papers, COMPCON 78, Spring, The Institute of Electrical and Electronics Engineers, Inc., New York, New York, 1978.

18. Putnam, Lawrence H., "Progress in Modeling the Software Life Cycle in a Phenomenological Way to Obtain Engineering Quality Estimates and Dynamic Control of the Process", Proceedings, Second Software Life Cycle Management Workshop, The Institute of Electrical and Electronics Engineers, Inc., New York, New York, 1978, pp. 105-128.

19. Putnam, Lawrence H., "The Real Economics of Software Development", Symposium On the Economics of Information Processing, IBM Systems Research Institute, 15-19 December 1980, pp. 801-818.

20. Putnam, Lawrence H., Software Cost Estimating and Life Cycle Control: Getting the Software Numbers, The Institute of Electrical and Electronics Engineers, Inc., New York, New York, 1980.

21. Putnam, Lawrence H.; Fitzsimmons, Ann, "Estimating Software Costs", Datamation, Volume 25, Number 10, September 1979, pp. 189-198; Volume 25, Number 11, October 1979, pp. 171-178; Volume 25, Number 12, December 1979, pp. 137-140.

22. Putnam, Lawrence H.; Wolverton, Ray W., Quantative Management: Software Cost Estimating, The Institute of Electrical and Electronics Engineers, New York, New York, 1977.

23. SLIM (an example of output from SLIM), Quantative Software Management, Inc., Undated.

24. SLIM Reference Notebook, Quantative Software Management, Inc., Undated.

25. SLIM User's Guide, Quantative Software Management, Inc., Undated.

26. United States General Accounting Office Report To the Congress: Contracting For Computer Software Development-Serious Problems Require Management Attention To Avoid Wasting Additional Millions, Report Number FGMSD-80-4, 9 November 1979.

27. Walker, W.H., IV, Approach to Software Life Cycle Cost Modeling, M.S. Thesis, Air Froce Institute of Technology, Wright-Patterson Air Force Base, Ohio, July 1979.

28. Walston, C.E.; Felix, C.P., "A Method of Programming Measurement and Estimation", IBM Systems Journal, Volume Sixteen, Nubmer One, 1977, pp. 54-73.

29. Wolverton, R.W., "The Cost of Developing Large-Scale Software", IEEE Transactions on Computers, Volume C-23, Number 6, June 1974, pp. 615-636.

BIBLIOGRAPHY

Barnum, K. Dorsey, Roadblocks to ADP Progress, Leadership and Management Development Center, Air University, Maxwell Air Force Base. Alabama, June 1979.

Cortada, James W., EDP Costs and Charges: Finance, Budgets, and Cost Control in Data Processing, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1980.

Costing Methods and Models For Acquisition Planning, Budgeting and Contracting, United States Army Logistics Management Center, Fort Lee, Virginia, April 1979.

Finfer, Marcia F.; Mish, Russell K., Software Acquisition Management Guidebook: Software Cost, Estimation and Measurement, System Development Corporation, Santa Monica, California, March 1978.

Herd, James H.; Postak, John N.; Russell, W.E.; Stewart, Kenneth R., Software Cost Estimation Study, Volume I: Study Results, RADC-TR-77-220-Volume I, Doty Associates, Inc. for Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base, New York, June 1979.

Howard, Dennis D., An Automated Support Costing System, Leadership and Management Development Center, Air University, Maxwell Air Force Base, Alabama, June 1979.

Jelinski, Zygmunt, Configuration Management and Software, Defense Systems Management Review, DSMC, Ft. Belvoir, Virginia, September 1979.

Morin, Lois H., Estimation of Resources for Computer Programming Projects, M.S. Thesis, University of North Carolona, Chapel Hill, North Carolins, 1974.

Naval Data Automation Command Life Cycle Management Guidebook, March 1981.

Putnam, Lawrence H., "Life-Cycle Management Measurement Models: Predictive", Proceedings, Second Software Life Cycle Management Workshop, The Institute of Electrical and Electronics Engineers, Inc., New York, New York, 1978, pp. 32-39.

Parr, F.N., "Alternative to the Rayleigh Curve Model For Software Development Effort", IEEE Transactions on Software Engineering, Volume SE-6, Number 3, May 1980, pp. 291-296.

Shooman, Martin L., Tutorial on Software Cost Models, Polytechnic Institute of New York, Farmingdale, New York, October 1979.

Sorkowitz, Alfred R., "Software Life Cycle Costing", Proceedings of Trends and Application 1979, Institute of Electrical and Electronics Engineers, Inc., New York, New York, 1979.

Steffy, R.E., Programmed Review of Information for Costing and Evaluation-Software---Analysis of the RCA Price-S Cost Estimation Model AS It Relates To Current Air Force Computer Software Acquisition and Management, M.S. Thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, October 1980.

Stone, Dr. Harold S., Life Cycle Cost Analysis of Instruction-Set Architecture Standardization For Military Computer Based Systems, University of California, Berkeley, California, July 1978.

Waina, R.B.; Foreman, G.L.; Bangs, A.P.; Green, J.L.; Rodriguez, E.E., Predictive Cost Model Study, Volume 1: Final Technical Report, Hughes Aircraft Company, Aerospace Systems Division, Canoga Park, California, June 1980.

Waina, R.B.; Bangs, A.P.; Rodriguez, E.E., Predictive Software Cost Model Study, Volume 2: Software Package Detailed Data, Hughes Aircraft Company, Aerospace Systems Division, Canoga Park, California, June 1980.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Defense Logistics Studies Information Exchange U.S. Army Logistics Management Center Fort Lee, Virginia 23801	2
3.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
4.	Acquisition Library, Code 54A1 Naval Postgraduate School Monterey, California 93940	1
5.	Department Chairman, Code 36 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
6.	Department Chairman, Code 54 Department of Administrative Sciences Naval Postgraduate School Monterey, California 93940	1
7.	Commander M.L. Sneiderman, USN, Code 54Zz Department of Administrative Sciences Naval Postgraduate School Monterey, California 93940	1
8.	Professor Norman R. Lyons, Code 54Lb Department of Administrative Sciences Naval Postgraduate School Monterey, California 93940	1
9.	Commander J. Pfeiffer, USN, Code 37 Naval Postgraduate School Monterey, California 93940	1
10.	Lieutenant Commander R. A. Bobulinski, USN, Code 54Bb Department of Administrative Sciences Naval Postgraduate School Monterey, California 93940	1

11. Lieutenant Commander Kenneth C. Clare, USN — 1
CNO (OP-942D)
Department of the Navy
Washington, D.C. 20350

12. Mr. Carl Day — 1
Headquarters, United States Marine Corps (CCIS)
Washington, D.C. 20381

13. Mr. Lawrence H. Putnam — 1
Quantitative Software Management, Inc.
1057 Waverly Way
McLean, Virginia 22101

14. Mr. Bill Ward — 1
TRW Defense and Space Systems Group
550-24th Street, Suite #202
Ogden, Utah 84401

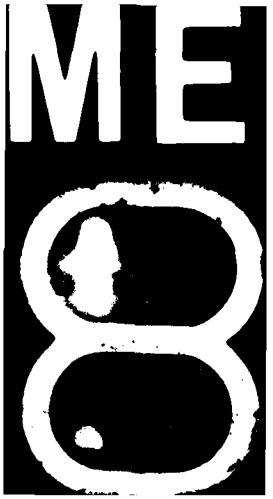
15. Mr. Harlan C. Chase — 1
Computer Sciences Corporation
2251 San Diego Avenue
San Diego, California 92110

16. Mr. S. P. Tomesek — 1
Vice President for Corporate Planning
Columbus and Southern Ohio Electric Company
215 North Front Street
Columbus, Ohio 43215

17. Captain Blair R. Vorgang, USMC — 2
2570 North Providence Road
Media, Pennsylvania 19063

18. Professor Dan C. Boger, Code 54Bk — 1
Department of Administrative Sciences
Naval Postgraduate School
Monterey, California 93940

19. Professor Michael G. Sovereign, Code 55Zo — 1
Department of Operations Research
Naval Postgraduate School
Monterey, California 93940


ME
8